UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Fernando B Cosentino

DESENVOLVIMENTO DE SISTEMA INTEGRADO CONVERGENTE DE MONITORAMENTO DE SINAIS BIOMÉDICOS USANDO MICROCONTROLADORES DE 8 BITS

Florianópolis

2013

Fernando B Cosentino

DESENVOLVIMENTO DE SISTEMA INTEGRADO CONVERGENTE DE MONITORAMENTO DE SINAIS BIOMÉDICOS USANDO MICROCONTROLADORES DE 8 BITS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica para a obtenção do Grau de Mestre em Engenharia Elétrica.
Orientador: Prof. Dr. Fernando Mendes de Azevedo
Coorientador: Prof. Dr. José Marino Neto

Florianópolis

2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Cosentino, Fernando Bruinjé
Desenvolvimento de sistema integrado convergente de monitoramento de sinais biomédicos usando microcontroladores de 8 bits / Fernando Bruinjé Cosentino ; orientador, Fernando Mendes de Azevedo ; coorientador, José Marino Neto. - Florianópolis, SC, 2013.
92 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Elétrica.

Inclui referências

1. Engenharia Elétrica. 2. Saúde ubíqua. 3. Telemedicina. 4. Sinais vitais. 5. Monitoramento. I. Azevedo, Fernando Mendes de. II. Marino Neto, José. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

Fernando B Cosentino

DESENVOLVIMENTO DE SISTEMA INTEGRADO CONVERGENTE DE MONITORAMENTO DE SINAIS BIOMÉDICOS USANDO MICROCONTROLADORES DE 8 BITS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia Elétrica", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica.

Florianópolis (SC), 23 de agosto de 2013

_____

Prof. Patrick Kuo-Peng, Dr.
Coordenador do Curso

Banca examinadora:

_____

Orientador: Prof. Fernando Mendes de Azevedo, Dr.
Universidade Federal de Santa Catarina

_____

Profª. Daniela Ota Hisayasu Suzuki, Dr.
Universidade Federal de Santa Catarina

_____

Prof. Jefferson Luiz Brum Marques, Ph.D.
Universidade Federal de Santa Catarina

_____

Profª. Giselle Lopes Ferrari Ronque, Drª
Universidade Federal do Paraná

_____

Profª. Fernanda Isabel Marques Argoud, Drª
Instituto Federal de Santa Catarina

AGRADECIMENTOS

Gostaria de prestar meus agradecimentos à Universidade Federal de Santa Catarina e ao Instituto de Engenharia Biomédica pela oportunidade de realizar este trabalho, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico pelo suporte, aos meus professores orientadores pelo auxílio, e aos colegas pelas sugestões.

RESUMO

O aumento da expectativa de vida combinada com a incidência de doenças crônicas tem sido um desafio aos sistemas de saúde. Sistemas de telemetria podem ser uma resposta, fornecendo recursos ubíquos. Diversas soluções incluem computadores de mesa operados pelo paciente ou profissional de saúde, enquanto outros usam dispositivos portáteis embarcados, freqüentemente com processadores ARM ou mais complexos, e um servidor *online* é uma escolha bastante comum. Nesse contexto, este trabalho apresenta um sistema para monitoramento de sinais biomédicos – principalmente sinais vitais – fortemente focado em baixo custo e flexibilidade em relação ao cenário. Ao invés de um processador central complexo, o sistema é construído inteiramente com microcontroladores de 8 bits, em topologia distribuída, modular e escalável. Módulos de aquisição podem medir sinais que possam ser representados como um único número atualizável (como freqüência cardíaca, temperatura ou pressão arterial), e podem ser em qualquer número até 100. Módulos de transmissão carregam essa informação em links de rádio ou físicos, um para cada módulo de aquisição. Um módulo principal lê os dados de todas as fontes e exibe em um servidor *web* embarcado. Toda a informação pode ser monitorada usando qualquer aparelho rodando um navegador que suporte *javascript*, e não é necessário que seja um computador. O sistema foi validado com sinais de referência, e os tempos de processamento foram medidos. Os resultados mostram como as taxas de dados variam com cada cenário, e foram elaboradas equações relacionando as características de execução do sistema ao número de módulos.

Palavras-chave: Engenharia biomédica. Telemedicina. Saúde Ubíqua. Monitoramento. Sinais biomédicos. Sinais vitais.

ABSTRACT

Medical telemetry is an attempt to reduce healthcare costs while providing ubiquitous features. Several solutions include desktop computers operated by the patient or health care professional, while others use wearable embedded devices, often with ARM or more complex processors, and an online server is a very common choice. In this context, this work presents a system for biomedical signals monitoring – vital signs in partiular – strongly focused on low-cost and flexibility over different scenarios. Instead of a complex central processor, the system is build entirely using 8-bit microcontrollers in a distributed, modular and scalable topology. Acquisition modules can measure signals that can be represented as a single updateable number (such as heart rate, body temperature and blood pressure) and can be in any number up to 100. Transmission modules carry this information over radio or physical links, one module for each acquisition module. One main module reads data from all sources and displays in an embedded webserver. All the information can be monitored using any device running a javascript-enabled browser, and no computers are required. The system was verified with reference signals, and processing times were measured. Results show how the data rates vary with each scenario, and equations have been found relating number of modules and respective reading times.

Keywords: Biomedical engineering. Telemedicine. U-Health. Monitoring. Biomedical signals. Vital signs.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Ao longo do último século, o desenvolvimento da medicina trouxe um aumento na expectativa de vida, e uma migração da causa de mortalidade, de doenças infecto-contagiosas para doenças não transmissíveis – como, por exemplo, as cardiovasculares (OMS, 1999). Esse aumento de enfermidades crônicas tem sido um desafio às capacidades dos sistemas de saúde. Em países europeus e nos Estados Unidos, as doenças cardiovasculares são as mais comuns, e a tendência é que sua proporção aumente (PARK et al., 2011). Atualmente, o desafio imposto globalmente aos sistemas de saúde pelo envelhecimento da população e a predominância de condições de longo prazo, já é plenamente estabelecido e conhecido (MURRAY et al., 2011).

Com o aumento da expectativa de vida, portadores de doenças e condições crônicas passarão mais tempo demandando estruturas de cuidado, aumentando os recursos necessários, no sistema de saúde.

O uso da Tecnologia da Informação (TI) para enfrentar esses desafios já é nítido (MURRAY et al., 2011), caracterizando, na área da saúde atual, um paradigma chamado de Saúde Ubíqua (u-Health).

Saúde Ubíqua é definida como serviços médicos e gestão de saúde, que utilizem redes com e sem fio, e são capazes de ser usadas em qualquer lugar ou tempo (KIM et al., 2011). Nela, dispositivos eletrônicos unem tecnologias de telemetria e telecomunicações para ampliar o alcance – no espaço e no tempo – de serviços médicos como, por exemplo, monitoramento de pacientes. Se esses serviços forem prestados com o paciente em casa – situação chamada de *home care* – a hospitalização pode ser reduzida significativamente, com conseqüente aumento de qualidade de vida e redução de custos (PARK et al., 2011).

Profissionais da saúde consideram que a acessibilidade e redução de tempo são também grandes vantagens da saúde ubíqua, e particularmente os profissionais envolvidos em diagnósticos consideram a habilidade de monitoramento à distância como a maior vantagem. Muitos pacientes moram longe de um hospital, e o transtorno de se deslocar até ele faz com que pacientes abandonem o tratamento (KIM et al., 2011), o que poderia ser evitado nos casos em que o motivo da visita possa ser resolvido à distância.

Se a comunicação dos dados entre paciente e profissional da saúde for feita via internet, ela poderá ser feita à curta distância (como paciente e profissional em salas diferentes de um mesmo edifício), e também à longa distância, como no caso de *home care*, abrangendo uma varie-

dade de casos com uma mesma tecnologia. Ao mesmo tempo em que permite a independência física do paciente, a internet também permite a independência física do profissional da saúde, uma vez que o uso de clientes *web* portáteis, como *smartphones*, tem crescido e adquirido importância na área de saúde (CHOI et al., 2011).

Uma forma de monitoramento que tem sido implementada é a em que o paciente se conecta a um computador periodicamente para digitar ou coletar dados. Em um teste realizado na Coréia, pacientes reclamaram da necessidade de ligar um computador e ficar conectado para realizar as medidas (PARK et al., 2011).

Para acesso à internet, o protocolo TCP/IP sobre Ethernet é uma combinação consolidada. Vários fabricantes de microcontroladores incluem em seus produtos soluções em hardware para sua implementação. Se um equipamento de monitoramento for construído utilizando um microcontrolador com Ethernet, informações poderão ser entregues via internet sem a necessidade de um computador, permitindo que o monitoramento esteja ativo o tempo todo com um menor consumo de energia e um custo menor por equipamento (caso ambos tenham que ser adquiridos).

Optando-se por uma estratégia descentralizada, separando a estrutura de monitoramento em partes com funções distintas comunicando-se sem fio, permite-se uma topologia em que o acesso a internet possa ser implementado em um único equipamento, enquanto os equipamentos de medida propriamente ditos podem ser reduzidos em tamanho, peso e custo, ficando em contato com o paciente (ou à pronta disposição) por períodos mais longos do que nas topologias em que o paciente deve se conectar a um computador. Caso seja necessário, o equipamento pode ser portátil e leve o suficiente para ficar em contato permanente com o paciente.

A escolha do conjunto específico de elementos para a convergência de tecnologias determinará as capacidades de medida do sistema como um todo, os limites de trabalho, escalabilidade e compatibilidade com outras tecnologias que possam ser adicionadas posteriormente.

Durante a pesquisa bibliográfica deste trabalho, foi constatado que nos trabalhos de propósito similar o desenvolvimento da parte de aquisição de dados têm recebido mais atenção do que a disponibilização dos dados em si, ou com arquitetura mais complexa que aquela.

O sistema de (TOLENTINO; PARK, 2010), de topologia semelhante a este trabalho, utiliza um microcontrolador AVR de 8 bits para o equipamento de leitura, e XBee para a transmissão dos dados. No entanto, embora proponha uma topologia para a recepção, não detalha seu hard-

ware. O sistema de (LEE et al., 2009) utiliza o hardware de um Nintendo DS, um videogame portátil baseado em microcontroladores ARM (7 e 9) e transmissor Wi-Fi, dispensando um equipamento receptor dedicado. O equipamento de (CHUNG, 2009) utiliza um microcontrolador de 16 bits da Texas Instruments, e IEEE 802.15.4 para a transmissão. O sistema de (JACOB et al., 2011) utiliza um laptop X0 para receber os dados de uma interface, disponibilizando em rede. O projeto proposto por (RAUT; GIRIPUNJE, 2011) utiliza microcontroladores de 8 bits e transmissão por rádio, mas não especifica a recepção.

A redução do custo e complexidade do equipamento de disponibilização dos dados – isto é, a parte com acesso à internet – pode colaborar para tornar esse tipo de solução mais viável e aumentar a penetração no sistema de saúde.

Microcontroladores de 8 bits possuem custo menor do que os de 16 bits ou ARM, e algumas empresas já possuem modelos com Ethernet implementada em hardware, como a linha PIC18FxxJ60, do fabricante Microchip. Se, por um lado, a utilização dessa arquitetura – mais simples – pode reduzir custos e complexidade, por outro, poderá impor restrições aos dados trafegados em relação a outras tecnologias, restringindo os casos em que possa ser aplicada.

1.1 OBJETIVOS

Investigar o uso de microcontroladores PIC de 8 bits para um sistema de monitoramento de sinais biomédicos convergindo comunicação Ethernet e wireless.

1.1.1 Objetivos Específicos

- Identificar uma topologia e conjunto de tecnologias, componentes e protocolos que tornem o objetivo viável;
- Investigar os limites de trabalho da solução proposta, e apresentar, sucintamente, as situações que são e que não são compatíveis com os limites identificados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SINAIS BIOMÉDICOS E SINAIS VITAIS

Sinais biomédicos são sinais usados para extrair informações a respeito do sistema biológico sob investigação (BRONZINO, 2000). Sinais vitais são medidas que contém informações críticas sobre o estado de saúde de um paciente. São eles: temperatura corporal, frequência respiratória, frequência cardíaca, pressão arterial, e – quando conveniente – saturação de oxigênio. Os sinais vitais podem ser usados para identificar a magnitude de uma doença, e marcam doenças crônicas. (THOMPSON, 2009)

A Tabela 1 mostra a faixa de freqüência para vários sinais biomédicos.

Tabela 1 – Sinais biomédicos e suas respectivas faixas de freqüência. (BRONZINO, 2000)

Sinal	Freqüência
Potencial de Ação	100Hz a 2kHz
Eletroneurograma	100Hz a 1kHz
Eletroretinograma	0,2Hz a 200Hz
Eletro-oculograma	0Hz a 100Hz
Eletroencefalograma de Superfície	0,5Hz a 100Hz
Eletrocorticograma	100Hz a 5kHz
Eletromiografia (fibra)	500Hz a 10kHz
Eletromiografia (superfície)	0,01Hz a 500Hz
Eletrocardiograma	0,05Hz a 100Hz

Para os sinais vitais, (THOMPSON, 2009) recomenda que as frequências respiratória e cardíaca sejam contadas por 30 segundos. Pressão arterial e temperatura são medidas uma única vez a cada verificação do estado do paciente, e não são processos instantâneos. Para a temperatura, um termômetro é deixado em contato com o paciente durante um intervalo de tempo, e a medida de pressão arterial é baseada na observação de pulsos cardíacos. São, portanto, sinais de baixa frequência de amostragem (ordem de Hertz).

2.2 TECNOLOGIAS PARA LONGA DISTÂNCIA - ETHERNET

Existem várias arquiteturas e protocolos de rede que podem ser usados em estabelecimentos assistenciais de saúde. Desses, a pilha TCP/IP sobre Ethernet é consideravelmente popular. Trata-se de um conjunto de protocolos operando conjuntamente, numa organização de camadas denominada *Internet Protocols Suite*, ou Suíte de Protocolos da Internet.

2.2.1 TCP/IP, Ethernet e o Modelo OSI

O modelo *Open Systems Interconnection* (OSI) é uma estrutura definida pela *International Organization for Standardization* (ISO) conjuntamente com a *International Electrotechnical Commission* (IEC), como padrão a ser adotado internacionalmente pelos serviços de comunicação entre sistemas computacionais. O modelo OSI divide os processos de telecomunicações em camadas, de forma que as características intrínsecas de uma camada sejam transparentes para as outras camadas, e que uma camada se baseie em serviços fornecidos pelas camadas abaixo (ISO/IEC, 1994). Quando dois sistemas se comunicam, de acordo com esse modelo, um protocolo de uma camada específica do primeiro sistema interage somente com a mesma camada do segundo sistema, carregando como carga útil os dados que formatam os protocolos aplicados acima dele, formando uma pilha de sete camadas, mostradas conforme a Figura 1.

Figura 1 – Estrutura de camadas do modelo OSI.

A *camada física* (*physical layer*) se refere aos meios físicos para a transmissão de bits, diretamente entre equipamentos fisicamente conectados. A *camada de enlace* (*data-link layer*) se refere aos protocolos de identificação e endereçamento dos equipamentos conectados pela camada física. A *camada de rede* (*network layer*) se refere aos protocolos de organização e roteamento da informação, interligando as conexões. A *camada de transporte* (*transport layer*) se refere aos protocolos que tornam a comunicação entre as duas pontas independente da estrutura da rede (isto é, a topologia da rede é transparente a partir da camada de transporte). A *camada de sessão* (*session layer*) se refere aos protocolos que organizam os dados transmitidos na forma de uma conversação. A *camada de apresentação* (*presentation layer*) se refere aos protocolos de formatação da informação, para que a representação dos dados seja mantida. E, por fim, a *camada de aplicação* (*application layer*) se refere aos protocolos que são o objetivo final da comunicação estar acontecendo. Isto é, a função útil que a transmissão cumpre.

O termo Ethernet se refere ao padrão IEEE 802.3, que contém especificações para a camada física, e para uma subcamada da camada de enlace chamada *Media Access Control* (MAC) (IEEE, 2008). A conexão Ethernet é uma das tecnologias sobre as quais é possível aplicar as tecnologias envolvidas numa rede internet.

Uma rede do tipo internet é construída sobre uma suíte de protocolos denominada *Internet Protocols Suite* (IP), e é definida no documento *Request for Comments* (RFC) 1122, publicado pela *Internet Engineering Task Force* (IETF). As conexões típicas na internet são conhecidas pelo termo TCP/IP, embora esse processo inclua outros protocolos.

Embora semelhante, a divisão de camadas da suíte não corresponde exatamente ao modelo OSI, sendo definida conforme a Figura 2.

A internet é, em si, independente do meio físico, existindo mais de uma alternativa para sua implementação. No caso específico sobre Ethernet, o padrão IEEE 802.3 especifica a camada física e atribui aos equipamentos um endereço de 48 bits, denominado *endereço físico*, a ser usado pelo protocolo MAC na camada de enlace.

Na camada chamada *Internet*, o *Internet Protocol* (IP), definido no documento IETF RFC 791, considera um endereço de 32 bits para cada equipamento, chamado *endereço IP*. Uma das formas de atribuir um endereço IP a um equipamento com endereço físico é através do *Dynamic Host Configuration Protocol* (DHCP), definido pelo IETF RFC 2131. A suíte utiliza o *Adress Resolution Protocol* (ARP), definido no

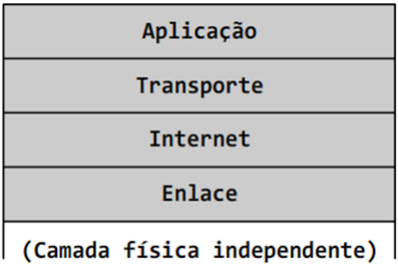

Figura 2 – Estrutura de camadas da suíte IP.

IETF RFC 826, para obter endereços físicos a partir de endereços IP. O protocolo IP utiliza o *Internet Control Message Protocol* (ICMP) para roteamento, diagnóstico e tratamento de erros, sendo definido no documento IETF RFC 792. O ICMP utiliza do próprio protocolo IP para transmitir suas mensagens, mas é considerado como parte intrínseca da camada, e não um serviço sobre ela. Essa camada também é chamada *camada IP*.

Na camada chamada *Transporte*, as mensagens são formatadas em um de dois protocolos: *User Datagram Protocol* (UDP) e *Transmission Control Protocol* (TCP). O UDP é um protocolo mínimo, permitindo que a aplicação utilize as características próprias do protocolo IP, isto é, pacotes de dados sem quaisquer garantias de recebimento ou ordem, adicionando somente a verificação de integridade e uma multiplexação por números chamados *portas*. O TCP, além dos mesmos recursos do UDP, lida com perda de dados e sequenciamento, permitindo que a informação seja disponibilizada de forma íntegra para as aplicações construídas sobre ele.

A camada chamada *Aplicação* engloba as funções que no modelo OSI estão nas últimas três camadas: *sessão*, *apresentação* e *aplicação*. Nesta camada funcionam protocolos como *Hypertext Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP), *Simple Network Management Protocol* (SNMP), e outros.

No caso específico do HTTP, definido em sua versão 1.1 no IETF RFC 2616, temos a estrutura conforme a Figura 3.

Definições de camadas mais altas do que o documento RFC 2616 são mantidas pela *World Wide Web Consortium* (W3C).

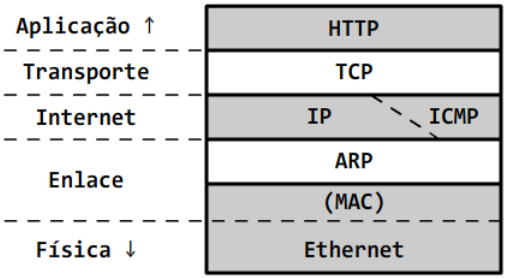

Figura 3 – Estrutura de camadas do serviço HTTP.

2.2.2 Pilhas TCP/IP Microchip

Como suporte ao *hardware* Ethernet incluído em seus produtos, a empresa Microchip fornece uma biblioteca de código que encapsula todo o processo de configuração e comunicação entre o *hardware* Ethernet e a aplicação. Dessa forma, o trabalho de implementação de uma comunicação TCP/IP em um microcontrolador com esse *hardware* se resume a um trabalho de configuração e manipulação da biblioteca, chamada *Pilha TCP/IP Microchip* ("Microchip TCP/IP Stack"). Essa pilha é escrita na linguagem C, específica para o compilador C18, este fornecido gratuitamente pelo fabricante do microcontrolador.

A biblioteca suporta os protocolos TCP e UDP, permitindo diversos serviços, como DHCP, HTTP, SMTP, FTP, e outros. A estrutura da pilha está mostrada na Figura 4.

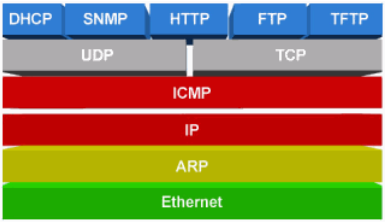

Figura 4 – Estrutura de camadas da pilha TCP/IP Microchip. (MICROCHIP, 2012)

O uso da biblioteca impõe restrições à aplicação:

- A pilha ocupa mais de 30KB, podendo passar de 100KB, dependendo dos serviços utilizados, logo o microcontrolador deve possuir memória de programa maior que isso;
- Utilizando HTTP, a aplicação deve contar com algum meio de armazenamento para as páginas que serão exibidas, seja esse meio EEPROM, FLASH ou externo (como um cartão SD);
- A biblioteca se baseia em uma tarefa (*task*) para lidar com a comunicação Ethernet, utilizando um *buffer* para armazenamento dos dados comunicados entre uma execução e outra da tarefa. Assim, essa tarefa deve ser chamada com periodicidade mais rápida possível, para evitar que o *buffer* se encha durante esses intervalos, impondo portanto uma restrição na duração do laço principal do programa. No entanto, o enchimento do *buffer* é diretamente dependente da quantidade de dados trafegados, não havendo, nas publicações do fabricante, quaisquer dados numéricos a respeito;
- A aplicação deve ser toda construída na forma de tarefas, que não bloqueiem ou atrasem o laço principal do programa (tarefas "non-blocking"), formando uma estrutura multitarefa ou um sistema operacional em tempo real. Para operações que usem *delays*, a biblioteca inclui uma contagem de tempo interna ("Tick") para ser usada ao invés de pausas lineares. Dessa forma, o tempo gasto na execução do laço principal é mantido mínimo, para evitar o enchimento do *buffer*.

Um programa em estrutura multitarefa, isto é, composto de tarefas *non-blocking*, consiste em blocos de instruções organizados por sua função, escritos para serem executados no menor tempo possível por chamada, mesmo que tenham que ser chamados várias vezes para completar as respectivas tarefas. Cada vez que uma função é chamada, ela executa somente um passo do seu processo, e retorna ao laço principal permitindo que a próxima função execute seu respectivo passo interno. Quando a função for chamada novamente na próxima iteração, executará o segundo passo, retornando em seguida, e assim sucessivamente. Dessa forma, um sistema com um único processador consegue executar várias tarefas de forma virtualmente paralela.

Utilizando a biblioteca numa estrutura multitarefa, toda a lógica necessária para a aplicação será executada de forma virtualmente paralela (apenas um passo de cada tarefa a cada execução do laço principal), permitindo que a tarefa específica associada à comunicação Ethernet seja chamada o mais rápido possível.

Para gerar atrasos que não bloqueiem a execução, as funções de *delay* devem utilizar o conceito de *ticks*: um *timer* do microcontrolador marca uma determinada contagem de tempo em uma variável, de forma que cada incremento é um *tick*. Para aguardar uma determinada quantidade de tempo, essa quantidade é convertida numa quantidade de *ticks*; no início da pausa, a tarefa armazena o valor de *ticks* decorridos até o momento, e passa a retornar sem ação alguma para o laço principal sempre que for chamada, até que o valor de *ticks* tenha sido incrementado naquele número calculado. Dessa forma, todas as outras tarefas são executadas normalmente enquanto aquela tarefa está em pausa.

2.2.3 *Hardware* Ethernet em microcontroladores Microchip de 8 bits

Dentre os microcontroladores PIC de 8 bits, o *hardware* dedicado à Ethernet está disponível na família PIC18, como por exemplo na linha PIC18FxxJ60, que possui modelos de microcontroladores munidos de:

- *Hardware* controlador Ethernet (IEEE 802.3), incluindo a camada física (PHY) e endereço físico (MAC) integrados;
- Freqüência de trabalho de até 41,667 MHz;
- 64, 80 ou 100 pinos;
- Memória de programa (FLASH) de 64KB, 96KB ou 128KB;
- 3808B de memória RAM;
- *Buffer* Ethernet de 8192B;
- Entre 11 e 16 conversores analógico-digitais (ADs) de 10 bits;
- Comunicação MSSP (SPI e I^2C) e EUSART (RS-485, RS-232 e LIN 1.2);
- *Timers* de 8 e 16 bits (2 de 8 bits e 3 de 16 bits);
- Alguns modelos possuem acesso a memória externa.

O fabricante fornece, gratuitamente, o compilador C18, compatível com essa linha de microcontroladores e com a biblioteca TCP/IP.

2.2.4 *Webserver*

Um *serviço web* (*WebService*) é um sistema de *software* desenvolvido para permitir interação máquina-a-máquina sobre uma rede. Outros serviços interagem com o serviço web tipicamente através do protocolo HTTP, aliado a outras tecnologias (W3C, 2004).

A principal atividade de um webserver é fornecer páginas para navegadores. Essas páginas podem ser geradas através de alguma linguagem de programação, ou estarem armazenadas em arquivos prontos, e podem ser de texto puro, ou escritas por linguagens compreendidas pelo navegador, como HTML (*HyperText Markup Language* – descreve a diagramação da página), *JavaScript* (comandos executados pelo navegador), XML (*Extensible Markup Language* – usada para organizar dados) e CSS (*Cascading Style Sheet* – usada para descrever ou modificar a aparência da página).

Essas linguagens são frequentemente combináveis, e são processadas em momentos diferentes. Uma combinação bastante utilizada é HTML com JavaScript embutido. Nesse caso, o servidor envia uma página em HTML, que contém códigos JavaScript em seu interior. O navegador interpreta todas as marcações HTML e exibe uma versão inicial da página. Após o carregamento e exibição, lê os códigos JavaScript contidos na página e os executa localmente. Esses códigos podem manipular o comportamento do navegador (redirecionando para outras páginas ou exibindo avisos, por exemplo), como podem alterar o conteúdo da própria página (inserindo, removendo ou alterando os elementos da linguagem HTML).

A técnica AJAX (*Asynchronous JavaScript and XML*) utiliza comandos JavaScript e dados organizados em XML para solicitar ao servidor informações em tempo real, sem que seja necessário atualizar a página HTML. Em uma requisição AJAX, uma conexão é aberta em segundo plano, e uma página é requisitada normalmente através do protocolo http. No entanto, ao invés de exibir como nova página, o navegador repassa o resultado para uma função JavaScript dentro da página original, que decidirá o que fazer com a informação. Dessa forma, o comportamento da página ganha agilidade, uma vez que mantém a estrutura original, substituindo apenas pequenos trechos com dados atualizados.

2.3 TECNOLOGIAS PARA CURTA DISTÂNCIA

Neste trabalho, foi considerado como curta distância o espaço pessoal da ordem de metros, como por exemplo a área dentro de uma sala.

2.3.1 UART

UART (*Universal Asynchronous Receiver/Transmitter*) é uma tecnologia com mais de quarenta anos de maturidade, de transmissão serial de bytes na qual os pacotes de dados consistem em um único byte. Toda a estrutura dos dados deve ser arbitrada nas camadas superiores da comunicação.

Os pacotes de dados são formados por um bit de início (*start bit*), uma quantidade de bits úteis (entre quatro e oito, tipicamente oito), opcionalmente um bit de paridade, para auxiliar a detecção de erros (se usado, o número de bits "1" deve ser sempre par), e um símbolo de encerramento (*stop bits*), que pode ser um bit, um bit e meio, ou dois bits (tipicamente um bit). Sendo assíncrona, a transmissão ocorre sem qualquer sinalização de *clock*, em uma velocidade fixa e conhecida por todas as pontas, denominada *baud rate*.

Em sua configuração mais típica, a comunicação é feita entre especificamente duas pontas, bidirecional em duas vias dedicadas a cada sentido de transmissão (configuração *full-duplex*), embora existam as possibilidades unidirecional e bidirecional em uma via (*half-duplex*), além de protocolos com mais de duas pontas (através de barramentos específicos de cada protocolo).

Para microcontroladores que possuem *hardware* UART integrado, a taxa de transmissão máxima é limitada pela velocidade do oscilador. Além disso, a velocidade é determinada através de bytes de configuração, que possuem uma faixa limitada de valores. Dessa forma, devido à própria natureza digital da configuração, essas velocidades assumem degraus discretos. Quando o oscilador não possui um valor múltiplo da taxa desejada, é necessário optar por uma velocidade que, embora não seja do valor exato buscado, seja suficientemente próxima. A diferença entre o valor desejado e o valor permitido pela configuração do microcontrolador determina um *erro de baud rate*. Quando o erro é suficientemente grande para que ocorra o deslocamento de meio bit até o fim do pacote, o receptor fica incapaz de interpretar o dado recebido (já que o centro do bit – seu ponto mais estável – passa a ser o

momento de transição de informação). Dessa forma, além do oscilador causar limitações de taxa de transmissão diretamente por sua velocidade, também causa por seu valor específico não-múltiplo, devido ao erro de *baud rate*.

2.3.2 SPI e I²C

Vários modelos de microcontroladores possuem um *hardware* periférico que é compatível tanto com o protocolo SPI quanto I2C.

O protocolo *Serial Peripheral Interface* (SPI), criado pela empresa Motorola, consiste em duas vias de comunicação serial, síncronas, operando em regime *full duplex*, isto é, uma via inteiramente dedicada à transmissão, e outra inteiramente à recepção. Os pacotes de dados consistem sempre em um único byte, com cada bit sinalizado por um pulso de *clock* em uma terceira via. O envio e a recepção ocorrem simultaneamente durante os pulsos de *clock*.

Um dos dispositivos necessariamente é o *mestre* da transmissão, sendo o que controla a linha de *clock*. Vários dispositivos podem compartilhar as mesmas linhas de dados, em barramento, desde que somente um dispositivo *escravo* esteja ativo por vez, além do mestre – os outros permanecendo em alta impedância. O mestre controla o dispositivo escravo ativo através de uma linha *chip select* (CS) dedicada para cada escravo. Um sinal digital alto na entrada CS de um escravo o faz sair do estado de alta impedância, conectando seu periférico às linhas do barramento.

O protocolo Inter-Integrated Circuit (IIC, ou I²C), criado pela empresa Philips, consiste em duas vias *half-duplex* (isto é, a via pode transmitir em ambos os sentidos, não ao mesmo tempo), uma usada principalmente para dados, chamada *Serial Data* (SDA), a outra principalmente para *clock*, chamada *Serial Clock* (SCL).

Todos os dispositivos estão conectados a cada uma das vias em barramento, através de transistores em configuração coletor aberto/dreno aberto, ou seja, podem apenas conectar a linha ao nível lógico baixo, ou desconectar-se dela (alta impedância). As linhas são externamente conectadas ao nível lógico alto através de resistores de *pull-up*. Dessa forma, qualquer dispositivo do barramento pode criar um sinal digital em uma linha (conectando ao nível baixo, ou permitindo que o resistor mantenha o nível alto), mas se dois dispositivos manipularem a linha de forma conflitante, não ocorrerá corrente entre os dispositivos.

Os dispositivos escravos são identificados através de endereços

fixos em *firmware*, e toda a transmissão ocorre por sessões. Em uma sessão, necessariamente um dispositivo deve ser mestre, e todos os outros atuam como escravos (embora possam atuar como mestres em outras sessões). O mestre inicia a sessão através de uma combinação específica chamada *start condition*, na qual a linha SDA é colocada em nível baixo enquanto a linha SCL está alta. A sessão é encerrada com uma combinação específica chamada *stop condition*, na qual a linha SDA é liberada (colocada em nível alto pelo resistor de *pull-up*) enquanto a linha SCL está alta. Durante toda a sessão, todas as mudanças de nível da linha SDA devem acontecer com a linha SCL em nível baixo (para não causar *start condition* ou *stop condition*), e a informação é transmitida em palavras de oito bits, cada bit marcado por uma subida e descida sucessivas da linha SCL. Um nono bit, chamado *acknoledgement* (ACK) é adicionado pelo dispositivo que recebe a informação, com significados variados.

A primeira palavra de cada sessão é transmitida pelo mestre, e contém o endereço (de sete bits) do dispositivo escravo a que o resto da transmissão se destina, e um bit indicando se é uma sessão de leitura ou escrita. Se houver um dispositivo com aquele endereço, ele transmite um bit ACK. Em seguida, o mestre transmite os dados caso seja uma operação de escrita, ou fornece *clock* para que o escravo transmita os dados, caso seja uma operação de leitura. Na operação de escrita, o escravo transmite o bit ACK para indicar que o dado foi recebido, e nas operações de leitura, o mestre transmite o bit ACK para indicar que requisitará mais dados em seguida. Em ambos os casos, a sessão dura até que o mestre cause uma *stop condition*, ou uma nova *start condition* (chamada de *repeated start*, com o mesmo efeito de encerrar a sessão e iniciar outra). Para que o mestre possa enviar e receber dados de um mesmo escravo – por exemplo, enviar um comando e obter uma resposta – deve iniciar uma sessão de escrita, enviar o comando, e então iniciar uma sessão de leitura e obter a resposta. Não é possível modificar a direção de transmissão da carga útil dentro de uma mesma sessão.

2.3.3 *Bluetooth*

O termo *Bluetooth* é a designação comercial de um protocolo de transmissão sem fio, originalmente proprietário, e atualmente sob o padrão IEEE 802.15.1. Destina-se a conectar equipamentos portáteis pessoais entre si, ou com computadores, através de ondas de rádio da banda pública de 2,4 GHz.

O protocolo permite uma grande variedade de funções, e os dispositivos *Bluetooth* utilizam perfis para identificar qual a função que desempenham. Um perfil *Bluetooth* é uma especificação de protocolos e recursos utilizados sobre os protocolos básicos *Bluetooth*.

Neste trabalho, os dispositivos *Bluetooth* utilizaram o *Serial Port Profile* (SPP), cuja função é simular uma conexão UART, tornando o enlace por rádio invisível para a aplicação.

2.3.3.1 *Firmware HC-05*

O HC-05 é um *firmware* que implementa sobre *Bluetooth* o perfil SPP, e adicionalmente fornece um conjunto de configurações. Essas configurações incluem, entre outras, a taxa de transmissão, o modo de operação (entre mestre e escravo), e um endereço vinculado – isto é, caso um segundo dispositivo com aquele endereço esteja transmitindo ao alcance, o dispositivo conecta-se a ele automaticamente. O dispositivo que buscará o endereço vinculado deve estar operando como mestre, e o dispositivo possuidor do endereço vinculado funcionará como escravo. O endereço de um dispositivo é um valor numérico fixado pelo fabricante, e único para cada exemplar (mesmo entre fabricantes diferentes).

Os dispositivos que utilizam o *firmware* HC-05 possuem um terminal dedicado a configuração: enquanto esse terminal estiver em nível lógico alto, toda a informação enviada através de sua porta UART será interpretada como configuração. Com o terminal em nível lógico baixo ou flutuante, a informação enviada pela porta UART será transmitida à outra ponta do enlace *Bluetooth*.

Os comandos de configuração relevantes neste trabalho estão listados na Tabela 2.

2.3.4 XBee

O termo XBee é o nome comercial, criado pela empresa Digi International, de uma família de dispositivos baseados no padrão IEEE 802.15.4, com *firmware* de configuração proprietário. São transmissores por rádio para distâncias pequenas (ordem de dezenas de metros), e baixa velocidade de transmissão (ordem de kBps), priorizando o baixo consumo de energia, para aplicações que exigem alta disponibilidade, porém baixo volume de dados (como automação e monitoramento).

Tabela 2 – Comandos de configuração dos módulos bluetooth utilizados neste trabalho.

Comando	Efeito
AT+ADDR?	Responde com o endereço deste dispositivo.
AT+ROLE=<número>	Configura o dispositivo como escravo (número = 0) ou mestre (número = 1).
AT+UART=<taxa>,<stop bits>,<paridade>	Configura os parâmetros de funcionamento da porta UART.
AT+CMODE=<modo>	Configura o modo de conexão. Conecta-se somente a um endereço vinculado (modo = 0) ou livremente (modo = 1).
AT+BIND=<endereço>	Define um endereço vinculado, para conexão automática.

Diferentemente do grande leque de perfis da tecnologia *Bluetooth*, a transmissão serial por UART é o foco central da tecnologia XBee. A faixa de freqüência também é a banda pública de 2,4GHz.

Os dispositivos XBee podem operar de duas formas diferentes: transparente, na qual dois dispositivos são conectados ponta-a-ponta, simulando uma conexão UART direta – *firmware* denominado pelo fabricante como AT – ou através de um protocolo proprietário, no qual as informações são enviadas em pacotes, e que permite acesso e configuração do dispositivo durante a transmissão – firmware denominado pelo fabricante como API. A rede pode assumir diversas topologias, envolvendo ou não, além dos dispositivos finais, dispositivos coordenadores e roteadores. A forma relevante para este trabalho é a transparente envolvendo apenas dispositivos finais.

Os dispositivos XBee utilizam a arquitetura de rede *ZigBee*, que é um padrão internacional. Essa arquitetura se organiza em PANs (*Personal Area Network*), em que um dispositivo transmite um identificador numérico da PAN a que ele pertence, e só é visto por outros dispositivos que estejam na mesma PAN.

Assim como os dispositivos *Bluetooth*, dispositivos XBee também possuem endereços numéricos e recursos de vínculo para conexão automática. No entanto, possuem duas formas de endereçamento: cada dispositivo possui um endereço de 64 bits definido durante a fabricação, e também um endereço de 16 bits que pode ser redefinido pela aplicação,

com o propósito de ser um endereço temporário dentro de uma determinada rede organizada por um dispositivo coordenador. Se o primeiro (64 bits) for utilizado para a conexão automática, o vínculo será fixo, especificamente entre aqueles dois exemplares, de forma inequívoca, e à parte de quaisquer redes coordenadas.

O fabricante fornece um software para alteração dos parâmetros de configuração. Os parâmetros relevantes neste trabalho estão listados na Tabela 3.

Tabela 3 – Comandos de configuração dos módulos XBee utilizados neste trabalho.

ID	Número identificador da rede PAN
DH	32 bits mais significativos do endereço de destino da transmissão
DL	32 bits menos significativos do endereço de destino da transmissão
SH	32 bits mais significativos do endereço próprio
SL	32 bits menos significativos do endereço próprio
BD	*Baud rate*, de acordo com uma lista fornecida em *datasheet*, de 0 = 1200bps até 7 = 115200bps

3 METODOLOGIA

A Figura 5 mostra o diagrama em blocos, simplificado, da topologia proposta para atender aos pontos levantados.

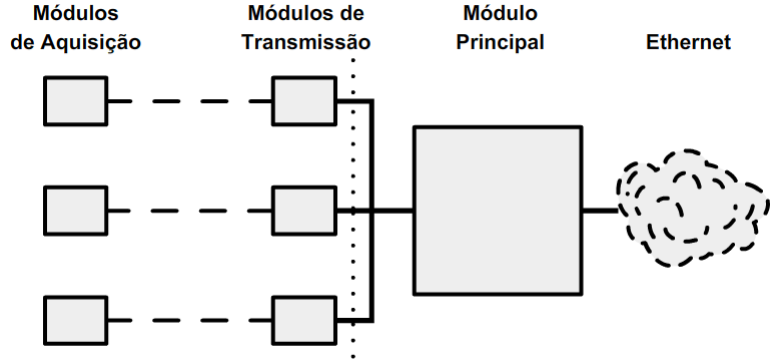

Figura 5 – Diagrama em blocos da topologia proposta.

O escopo deste trabalho é a transmissão e organização da informação, não envolvendo a aquisição física dos sinais nem o equipamento específico do usuário. A Figura 6 mostra todo o fluxo da informação, da origem ao usuário, marcando em linha tracejada o trecho abrangido pelo sistema proposto.

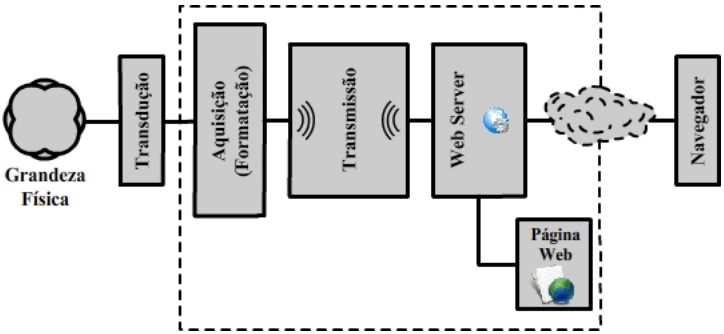

Figura 6 – Fluxo da informação, da fonte ao usuário. A linha tracejada indica o trecho abrangido pelo sistema proposto.

3.1 MATERIAIS

3.1.1 Microcontrolador com Ethernet

Para o módulo central, dotado de comunicação Ethernet, foi escolhido o microcontrolador PIC18F87J60, montado na placa modelo XBEENET do fabricante Microgenios. Além do microcontrolador, essa placa contém:

- Cristal de 25 MHz para *clock* central;
- Cristal de 32,768 kHz para *Timer1*;
- Conector RJ45 com transformador integrado;
- Memória serial EEPROM 24LC512;
- Conectores DB9 e MAX3232 para conexão RS-232 em duas portas;
- Display LCD compatível com HD44780;
- Conector para cartões SD;
- Dois relés com, respectivamente, dois transistores;
- Dois trimpots;
- Cinco push-buttons;
- Oito LEDs.

A comunicação Ethernet foi realizada baseada na biblioteca *Microchip TCP/IP Stack*. Foram implementados os itens da biblioteca necessários para o serviço HTTP. Foi utilizado o compilador C18, e o ambiente de desenvolvimento MPLAB, ambos fornecidos pela empresa Microchip.

Para gravação do *firmware* no microcontrolador, foi utilizado o gravador MicroICD, da empresa Microgenios, e o software PICkit2, da empresa Microchip.

Além da biblioteca, a Microchip fornece exemplos de códigos para o programa principal, parcialmente ou plenamente implementados, para que o programador modifique e adapte à aplicação, ao invés de escrever o programa principal do início. Este trabalho se baseou no

código "TCPIP Demo App", removendo as implementações dos protocolos SMTP, SNMP, Telnet, SNTP (*Simple Network Time Protocol*) e os serviços de ponte UART-TCP e DNS dinâmico, uma vez que este trabalho se baseia no serviço HTTP somente.

3.1.2 Demais Microcontroladores

Para todos os outros pontos da topologia ou do estudo em que foi necessário algum processamento em *hardware*, foram usados microcontroladores do mesmo fabricante e arquitetura, para eliminar a possibilidade de incompatibilidades entre fabricantes, e para restringir a fundamentação teórica de arquiteturas de microcontroladores e o respectivo *know-how*. Foram escolhidos os microcontroladores PIC16F628A e PIC16F73, pertencendo a uma família de complexidade menor que a PIC18F, possuindo os recursos necessários com menor custo.

Esses microcontroladores, não possuindo a dependência do compilador C18 imposta pela biblioteca TCP/IP, foram programados utilizando a linguagem Pascal e o compilador MikroPascal PRO, da empresa MikroElektronika. Em alguns casos, bibliotecas proprietárias da MikroElektronika, integradas ao compilador, foram utilizadas para brevidade de código e tempo.

Para gravação, foi utilizado o gravador Picburner USB, nome comercial de uma implementação do projeto aberto Pic Brenner8. Foi utilizado o software US-Burn.

3.1.3 Rádios

Para a comunicação sem fio entre os microcontroladores, foram escolhidas alternativas compatíveis com a comunicação UART, que é comum a todos os microcontroladores deste trabalho. Dentre essas, foram escolhidas duas alternativas.

3.1.3.1 *Bluetooth*

Para a comunicação *Bluetooth*, foram usados módulos conversores *Bluetooth-Serial* com o BC417143, um microcontrolador da linha BlueCore4 da empresa *Cambridge Silicon Radio* (CSR).

O *firmware* utilizado foi o HC-05, que implementa um *Serial Port Profile* (SPP), permitindo que um par de módulos simule uma

conexão UART direta.

Os módulos foram configurados com endereço vinculado dois a dois, formando canais UART transparentes e automáticos, na velocidade de 19200 bps. Assim, a configuração foi feita somente uma vez, sem a necessidade de qualquer acesso a elas por parte dos microcontroladores.

3.1.3.2 XBee

Para a comunicação XBee, foram utilizados módulos MaxStream, da empresa Digi International. O modo de comunicação utilizado foi o AT, e os dispositivos foram configurados em pares, isto é os parâmetros de configuração SH e SL de um foram configurados como DH e DL, respectivamente, no outro, e vice-versa. Dessa maneira, forma-se um canal UART transparente e automático. O parâmetro BD foi ajustado em 4, indicando baud rate de 19200 bps

3.2 FERRAMENTAS DESENVOLVIDAS

No decorrer deste trabalho, mostrou-se necessário efetuar medidas não contempladas nos instrumentos disponíveis. Dessa forma, ferramentas foram construídas para essas medidas, que não integram o protótipo em si, mas que fazem parte da metodologia e são também, portanto, contribuições deste trabalho.

3.2.1 Contador de Pacotes

Para as situações em que foi necessário uma contagem de pacotes (bytes) transmitidos em uma porta serial RS-232, foi elaborado um programa contador de bytes recebidos, executado na placa XBEENET. Foi utilizado para verificar a perda de pacotes transmitidos pelos módulos de rádio, dentro do alcance especificado pelos respectivos manuais.

Essa ferramenta consiste em uma porta serial EUSART, um display LCD, duas teclas (*push-buttons* rotulados na placa como S6 e S2) e o microcontrolador PIC18F87J60. O diagrama é mostrado na Figura 7.

Para melhor organização de código, a contagem foi organizada em blocos de 50.000 bytes. O display mostra o número de blocos completos recebidos, e o número de bytes recebidos que ainda não comple-

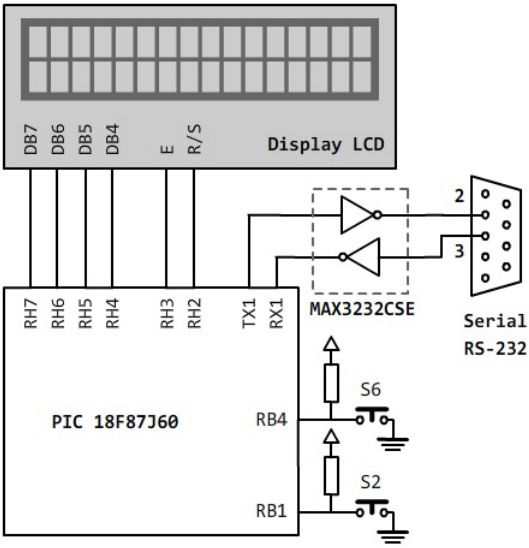

Figura 7 – Diagrama esquemático do contador de pacotes, implementado sobre a placa XBEENET.

taram um bloco.

Duas formas de sinalização, através de um dos pinos do microcontrolador, foram implementadas. A primeira indica o tempo entre bytes: o estado de um pino de saída é alternado cada vez que um byte é recebido.

A segunda sinalização indica o tempo de duração de um bloco. Essa informação pode ser usada para se obter a média do tempo entre bytes, uma vez que o tamanho do bloco é fixo e conhecido. Dessa forma, os erros pontuais são diluídos, os aleatórios se cancelam, e a resolução do instrumento de medida de tempo é melhorada em 50.000.

3.2.2 Cronômetro e Capturador de Eventos

Para medidas de tempo curtas (de microssegundos e milissegundos), foi utilizado um osciloscópio. No entanto, para medidas de tempo de intervalos maiores (segundos, minutos ou horas) com resolução da ordem de milissegundos e sem interferência humana, foi desenvolvido um cronômetro, utilizando um PIC16F873, com captura de um sinal digital. Foi utilizado para medir os tempos de transmissão de volumes conhecidos de dados pelos módulos de rádio.

O cronômetro consiste em uma placa com um cristal oscilador principal de 20MHz, para o programa; um cristal de 32768 Hz como *clock* externo de um *timer*, e uma porta USART. O diagrama do circuito é mostrado na Figura 8.

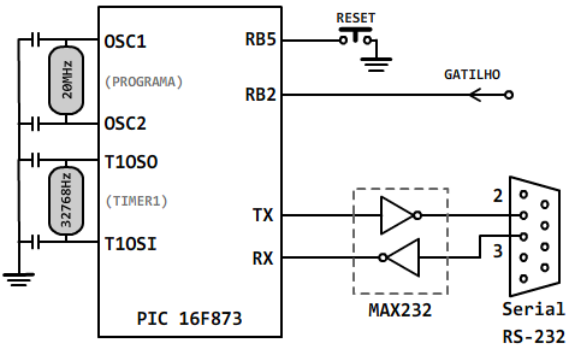

Figura 8 – Diagrama esquemático do cronômetro, implementado em placa própria.

O *Timer1* e o cristal de 32768 Hz foram configurados de forma a causar estouro na freqüência exata de 1 Hz (assumindo cristal ideal), funcionando como *Real Time Clock* (relógio em tempo real). Procurou-se sempre que possível comparar intervalos que tenham sido todos medidos com o mesmo cronômetro, para cancelar desvios sistemáticos causados pela não idealidade do cristal.

Baseada na contagem de segundos da interrupção, três variáveis de um byte, para hora, minuto e segundo, são usadas para marcar o tempo decorrido. O programa também tem acesso a uma variável de dois bytes, correspondendo ao número de *clocks* que o *Timer1* já recebeu desde o último estouro. Essa variável corresponde ao tempo, em unidade de 1/32768 de segundo, decorrido desde o último segundo inteiro contado. Essas quatro variáveis (hora, minuto, segundo e *clocks* do *Timer1*) são a saida do cronômetro.

O cronômetro possui uma única entrada digital (gatilho), que é monitorada ininterruptamente. Sempre que uma mudança no nível lógico do gatilho for identificado, o valor do relógio interno é enviado pela porta serial do microcontrolador.

3.3 MÉTODO

3.3.1 Tipos de Dados

Não tendo sido encontrados na literatura quaisquer padrões instituídos para a organização dos dados, foi necessário elaborar, neste trabalho, uma classificação própria para os tipos de dados medidos e transmitidos ao longo do sistema. Foram considerados e classificados três tipos de dados mensuráveis, tendo como critério conceitual a forma de amostragem e exibição da informação. Essa classificação independe da natureza física do sinal.

- **Estado**: uma informação pontual, cujo interesse é conhecer o valor atualizado (isto é, o mais recente), foi considerada aqui como um estado. As informações desse grupo possuem, como característica comum, o fato de que a chegada de uma nova informação descarta a informação anterior (quando se considera uma taxa de amostragem da ordem de Hertz ou mais). Um exemplo seria a temperatura: conhecendo o valor de temperatura corporal num determinado instante, o valor anterior dessa temperatura com alguns milissegundos de diferença pode ser descartado.

- **Seqüência**: uma informação que não está contida em um valor único, mas em uma continuidade de dados com evolução no tempo, como por exemplo uma forma de onda, foi considerada aqui como uma seqüência. As informações desse grupo possuem, como característica comum, o fato de que todas as amostras são necessárias para se reconstituir o sinal e apresentar a informação, com uma taxa de amostragem constante. Um exemplo seria o sinal de eletrocardiograma.

- **Evento**: uma informação que é pontual, no entanto que não pode ser atualizada a qualquer momento, só sendo válida em um determinado instante imprevisível, foi considerada como um evento. É uma informação que indica uma ocorrência, e cujo interesse é saber se ocorreu e quando ocorreu. Um exemplo seria a queda de um paciente.

3.3.2 Sinais e Medidas de Tempo

3.3.2.1 Medidas em Osciloscópio

Para as medidas de tempo até a ordem de milissegundos, foi utilizado um osciloscópio Tektronix TDS2002, 1GS/s. Três formas de medida de tempo foram adotadas, dependendo da informação desejada.

- **Medida direta**: é a medida da duração de um único evento, capturado e congelado. Os próprios cursores do osciloscópio foram usados para determinar o valor do intervalo.

- **Persistência**: utilizando a função persistência do osciloscópio, vários pulsos de uma seqüência periódica podem ser sobrepostos. Dessa forma, variações de duração podem ser identificadas numa mesma imagem.

- **Média**: utilizando a função média do osciloscópio, a informação mostrada é média de um determinado número de amostras. Com isso, pulsos ou picos de duração ou espaçamento variados podem ser identificados junto com suas proporções de ocorrência. Como a média é de um número determinado de amostras, funciona como uma média móvel. Assim, procurou-se usar a média em conjunto com a função persistência, para identificar a faixa total ocupada pela média móvel ao longo do tempo.

A Figura 9 mostra exemplos de duração e defasagem de pulsos, em imagens de persistência e média. Na imagem de duração de pulso por persistência, o ponto **a** indica o pulso mais curto, enquanto o ponto **b** indica o mais longo.

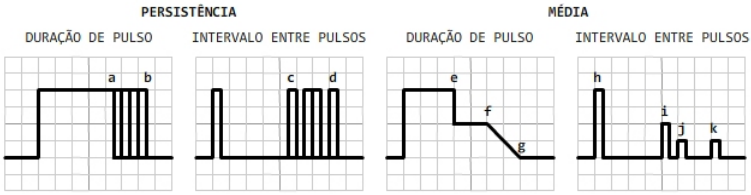

Figura 9 – Recursos de persistência e média do osciloscópio como indicadores limítrofes e estatísticos. Informações hipotéticas como exemplo.

Na imagem de intervalo entre pulsos por persistência, o ponto **c** indica o pulso que ocorreu após menor intervalo, enquanto o ponto **d** indica maior intervalo. Em ambas são mostrados registros intermediários, ignorados. O recurso de persistência representa a informação limítrofe.

Nas imagens por média, para pulsos digitais (sempre a mesma amplitude para o bit alto), a amplitude do primeiro ponto após a rampa de subida pode ser interpretada como a amplitude máxima, isto é, o topo de escala estatístico. Na imagem exemplo de duração de pulso por média, todos os pulsos duraram no mínimo até o ponto **e**, metade dos pulsos durou no mínimo até o ponto **f**, e para cada ponto a partir de **f**, a quantidade de pulsos que durou até ali caiu linearmente, até o ponto **g**, em que nenhum pulso alcançou. Na imagem de intervalo entre pulsos por média, metade dos pulsos ocorreu após um intervalo entre **h** e **i**; um quarto dos pulsos ocorreu após o intervalo entre **h** e **j**; e um quarto ocorreu após um intervalo entre **h** e **k**. Dessa forma, a imagem de média representa uma distribuição estatística, em relação à amplitude do primeiro ponto medido.

A Figura 10 mostra um exemplo detalhado de interpretação de duração de pulso por média.

Embora todos os pulsos tenham durado no mínimo até T1, uma determinada quantidade de pulsos durou apenas até T1. A proporção entre esses pulsos e o total de pulsos é dada pela razão **a**/**h**. Similarmente, a razão **b**/**h** indica a proporção de pulsos que durou especificamente até T2, **c**/**h** até T3 e **d**/**h** até T4.

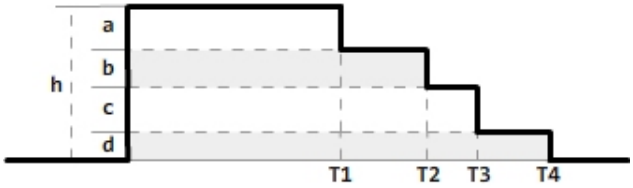

Figura 10 – Exemplo de duração de pulsos por média. As razões de **a**, **b**, **c** e **d** por **h** indicam as proporções de pulsos que duraram até T1, T2, T3 e T4, respectivamente.

3.3.2.2 Duração de Rotinas

Devido à natureza virtualmente paralela das tarefas executadas pelo microcontrolador conectado à internet, foi necessário medir o tempo levado por uma execução do laço principal. Para isso, um pulso foi criado em uma saída digital a cada execução do laço, formando trem de pulsos cujo período é a duração do laço, incluindo o tempo usado para criar o pulso, e o período negativo é a duração do laço sem o pulso.

Para capturar as variações ao longo do tempo, essa duração foi medida em osciloscópio, tanto por persistência como por média. Para evitar enchimento de *buffers*, a informação relevante é a condição limítrofe de maior duração.

A tarefa responsável pela comunicação Ethernet possui, em si, duração variável dependendo da carga armazenada pelo *buffer* (isto é, dependendo das conexões recebidas no momento), e é executada todas as vezes que o laço é percorrido. Dessa forma, foi necessário medir a duração do laço sem e com requisições para o servidor web. Para a duração com requisições, a medida foi feita com a situação normal de trabalho com um único visitante visualizando os dados, isto é, uma chamada AJAX a cada 100ms aproximadamente (variações se devem ao processamento do *javascript* pelo navegador, não podendo ser medidas).

Onde foi necessário medir o tempo entre duas comunicações sucessivas, uma saída digital foi alternada no início de cada comunicação, sendo o tempo entre duas transições de estado lógico o tempo entre dois processos de comunicação. No caso de comunicações feitas uma vez por execução de laço principal, esse tempo é idêntico à duração do

laço.

3.3.3 Sinais Gerados

Onde foi necessário gerar pulsos periódicos, foi utilizado um equipamento gerador de funções GoldStar FG-2002C, configurado na função *onda quadrada*, com saída TTL.

3.4 DESENVOLVIMENTO E ESTUDO DE CASO

3.4.1 Contexto

Buscou-se uma alternativa que pudesse ser implementada, com a menor adaptação possível, em dois contextos distintos: a residência de um paciente em *home care*, e um estabelecimento assistencial de saúde.

Em uma situação domiciliar, propõe-se que o paciente use um dispositivo portátil, vestido (por exemplo, ao pulso), para as medidas necessárias em tempo real; e um dispositivo posicionado num local de fácil acesso (por exemplo, sobre uma estante), a que o paciente se dirige e opera para as medidas necessárias apenas em intervalos regulares. O equipamento central é instalado em um ponto da residência, escolhido dependendo das características construtivas e dos hábitos do paciente, recebendo os sinais emitidos pelos pontos de medida. Como esse equipamento deve estar conectado a uma rede Ethernet, a disponibilidade de um cabo Ethernet ligado a uma rede intranet ou internet é uma exigência para a instalação do sistema. Uma vez que o equipamento está instalado na própria rede doméstica do paciente, este possui acesso aos próprios dados diretamente. O acesso de terceiros, através da internet, deve ser configurado nos equipamentos roteadores da rede, conforme desejado. Um exemplo está representado na Figura 11.

Nessa situação, várias medidas podem ser apresentadas pelo mesmo sistema, mas todos os dados se referem ao mesmo paciente.

Na situação de um estabelecimento assistencial de saúde, as informações podem se referir a várias medidas de um mesmo paciente, ou a uma mesma medida de vários pacientes, ou ambos simultaneamente. Dessa forma, é necessário que as informações sejam definidas em grupos, e que cada ponto de medida tenha, além da designação individual, a identificação de a que grupo pertence. A definição de cada grupo e elemento é uma característica de cada caso implementado. Por exem-

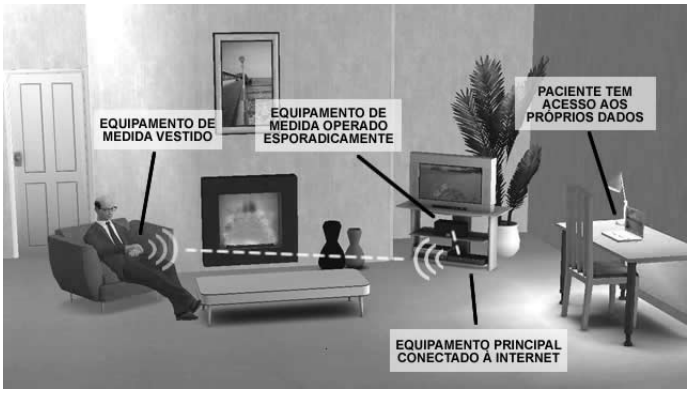

Figura 11 – Instalação proposta numa situação de *home care*.

plo, um estabelecimento pode considerar relevante exibir as informações agrupadas por paciente, e dessa forma *grupo* seria o paciente, e cada *elemento* seria cada medida. Outro estabelecimento pode considerar relevante o inverso, de forma que *grupo* seria uma medida, em que cada *elemento* seria um paciente.

Dependendo das características construtivas do estabelecimento, vários sistemas podem instalados simultaneamente (por exemplo, em locais fisicamente diferentes). Uma das preocupações deste trabalho foi a de que pacientes podem entrar e sair do sistema (seja por entrada e saída no estabelecimento, seja porque foram movidos de um local em que operava um sistema, para outro ao alcance de outro sistema), e isso deve ser possível sem que o sistema seja desligado para reconfiguração. Em outras palavras, adição e remoção de novos pontos de medida não devem interromper ou comprometer o funcionamento de outras medidas.

Considerando-se que a medicina é uma ciência em constante evolução, o conjunto de medidas específicas consideradas relevantes pode mudar com as conclusões de novas pesquisas. Dessa forma, este trabalho buscou uma abordagem flexível, em que o funcionamento do sistema seja o mais independente possível das grandezas medidas, e que a substituição ou aprimoramento da informação ou forma de medida seja até certo ponto possível sem necessidade de redesenvolvimento do sistema. Uma outra questão considerada é o fato de que diferentes pacientes possuem diferentes necessidades de monitoramento, sendo importante que o sistema possa ser implementado com foco no caso

particular, sem que todas as possibilidades de leitura tenham que ser desnecessariamente instaladas.

Por fim, foi considerada a estratégia de problema isolado: se, por qualquer interferência, uma determinada parte do sistema sofrer algum problema – como por exemplo um fio rompido, travamento de *firmware*, ou bateria esgotada – as demais partes não devem ser afetadas. Além disso, o problema deve ser, quando possível, solucionado pela própria rotina do sistema, através de protocolos internos sem exigir interferência humana. Se for necessária a reinicialização de algum *firmware*, ela deve ocorrer automaticamente e não deve exigir reinicialização de partes que não estejam relacionadas.

3.4.2 Sinais Medidos

Embora a flexibilidade seja uma característica buscada, é necessário delimitar um conjunto de sinais como foco. Para isso foram escolhidos os sinais vitais:

- Temperatura corporal;
- Frequência cardíaca;
- Pressão arterial;
- Freqüencia respiratória.

3.4.3 Topologia e Estrutura

Devido a todo esse contexto, a topologia proposta neste trabalho é descentralizada em estrutura multi-estágio e modular, conforme mostra a Figura 12.

O sistema possui três estágios. O primeiro estágio é responsável pela aquisição da medida, e formatação da informação. Esse estágio é modular: a cada informação medida, corresponde um *módulo de aquisição* (MA). Esse módulo pode ser ele mesmo um instrumento (lendo a informação diretamente do meio físico) ou receber dados de um outro instrumento, atuando como um conversor (essa forma é considerada prevendo compatibilidade com outros equipamentos). Em ambos os casos, cabe a esse módulo formatar a informação em uma mensagem que seja independente das peculiaridades do módulo, permitindo que os outros estágios lidem com os dados independentemente do tipo

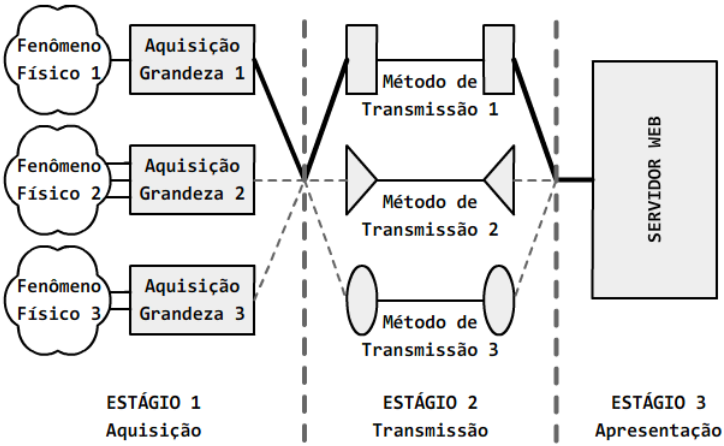

Figura 12 – Estrutura multi-estágio modular. Nos primeiro e segundo estágios, são mostradas três supostas alternativas para cada. Os números dos módulos são denominações hipotéticas, e a escolha da alternativa de um estágio não está associada à de outro estágio. A linha escura indica uma das possíveis escolhas.

de módulo de aquisição que a originou. Também cabe a esse módulo fornecer quaisquer informações a respeito do sinal, como identificação e morfologia.

O segundo estágio é responsável pela transmissão da informação, do módulo de aquisição até o equipamento principal onde está o servidor web. As entradas e saídas desse estágio são definidas: a entrada é a saída dos módulos de aquisição, padronizada independendo da grandeza medida; a saída é a entrada do equipamento principal, também padronizada, mas não necessariamente idêntica à entrada. Em outras palavras, identificar se há necessidade ou conveniência em converter e entregar a informação em formato diferente do recebido faz parte da investigação deste trabalho, assim como os protocolos entre os módulos, em si. Esse estágio é modular: o conjunto de todos os circuitos e equipamentos responsáveis por esse estágio é considerado o *módulo de transmissão* (MT). O módulo é transparente à entrega das mensagens: trocar um módulo de transmissão por outro de tecnologia de transmissão completamente diferente – desde que cumpra com as taxas de transmissão – não afetará em nada o funcionamento do sistema em relação aos dados.

O terceiro estágio é responsável pela recepção das informações, armazenamento e apresentação na forma de uma página web, incluindo as tarefas necessárias para atuar como um servidor HTTP. Também é responsável por toda eventual organização da comunicação, atuando como mestre de todos os dispositivos. É considerado o *módulo principal*, e consiste no equipamento central da topologia.

O terceiro estágio é hierarquicamente superior aos demais, e o segundo estágio é hierarquicamente superior ao primeiro. Em caso de falha em um módulo de aquisição, que cause travamento de *firmware*, ele é reiniciado por um WDT (*Watch Dog Timer*). Caso o *firmware* não esteja travado, permitindo comunicação normal com os outros estágios, mas seja identificada operação anormal, o respectivo módulo de transmissão pode controlar o módulo de aquisição, inclusive solicitando reinicialização. O módulo de transmissão também pode reiniciar o *hardware* de rádio associado a ele, e ao próprio *hardware*, caso não consiga se comunicar com o módulo de aquisição. O módulo principal pode solicitar todas essas operações, e também controlar o *buffer* de dados do módulo de transmissão.

3.4.4 Comunicação

A única forma de comunicação comum às formas de transmissão consideradas é a UART. Assim, a primeira forma de comunicação cogitada foi utilizar puramente essa tecnologia. No entanto, baseando-se nos dados mostrados na seção 4.1, conforme explicado na seção 5.1, essa forma foi considerada inviável.

O microcontrolador principal (PIC18F87J60) possui uma porta SPI/I^2C facilmente acessível, que também está disponível nos PIC16F873. Assim, o método optado foi o uso de um *buffer* de transmissão intermediário no módulo de transmissão, descarregado por I2C. A comunicação é feita por UART entre os módulos de aquisição e transmissão, até o *buffer* de transmissão, e por I^2C entre o *buffer* de transmissão e o módulo principal.

Por simplicidade de *hardware*, foi definido que a comunicação entre o primeiro e segundo estágio, em UART, obedece níveis lógicos na tensão de trabalho dos microcontroladores envolvidos. Há um protocolo de configuração entre os módulos de aquisição e transmissão, mas não há protocolo de requisição durante a transmissão de carga útil: o módulo de aquisição continuamente envia dados para o módulo de transmissão, em velocidade fixa. O canal pode permanecer ocioso por tempo indeterminado, no entanto quando não estiver ocioso a taxa de transmissão deve ser constante e conhecida. O protocolo de configuração permite ao módulo de transmissão enviar requisições de controle, solicitando informações de identificação ou enviando ordens de suspender dados ou reiniciar.

Não foram utilizados em nenhum momento protocolos formatados em texto (como XML), pois os bytes transmitidos para a formatação ocupariam um tempo de transmissão que poderia, de outra forma, ser usado para carga útil, uma vez que essa comunicação não incluirá equipamentos de outros sistemas nesse trecho. Assim, a definição do protocolo é de mais baixo nível.

As mensagens transmitidas para um nível hierárquico mais baixo (do módulo principal para o módulo de transmissão, e do módulo de transmissão para o módulo de aquisição) são consideradas *comandos*, e as transmitidas para níveis mais altos (do módulo de aquisição para o de transmissão, e do módulo de transmissão para o principal) são *respostas*. A comunicação entre os módulos de aquisição e transmissão é assíncrona, e entre os módulos de transmissão e principal é síncrona e controlada pelo módulo principal, ocupando a posição de Mestre I^2C. Todas as mensagens trocadas são na forma *comando-resposta*, neces-

sariamente juntos e necessariamente nessa ordem, exceto o envio de dados de medida do módulo de aquisição para o de transmissão, que consiste de um comando de início e múltiplas respostas.

3.4.4.1 Comunicação entre Primeiro e Segundo Estágios

A comunicação acontece ponta-a-ponta, por UART. O limite da velocidade da transmissão é imposto pelo menor *clock*, e a opção mais limitante (pior caso) é o oscilador interno do 16F628, no valor de 4MHz. Com esse valor, a taxa de transmissão mais alta com erro de *baud rate* tolerável (devido a *clock* não-múltiplo das taxas padrão) é de 19200bps. Assim, essa foi a velocidade implementada.

O módulo de aquisição possui três estados ("modos"): configuração, medição e desativado. No modo de configuração, são trocadas informações de identificação e controle. O módulo de aquisição não adquire medidas nem transmite dados, somente respondendo a requisições do módulo de transmissão. Nesse modo são recebidas, inclusive, as mensagens solicitando que mude de modo. Após a inicialização (*reset*), o módulo entra inicialmente em modo de configuração.

Ao receber instrução para começar a transmitir dados, o módulo entra em modo de medição. Nesse modo, transmite os dados das medições ininterruptamente, sem aguardar quaisquer requisições. As únicas informações recebidas do módulo de transmissão são uma confirmação de recebimento para cada amostra, e opcionalmente requisições de mudança de modo, que quando presentes são esporádicas. Dessa forma, com um pacote de dados composto de três bytes e um byte de confirmação, atinge-se a máxima taxa de transferência de 480 amostras por segundo.

O módulo entra em modo desativado quando recebe dados que não são coerentes com o protocolo de comunicação com o módulo de transmissão, assumindo que está conectado a um equipamento danificado ou um equipamento de outra natureza. Nesse modo, ignora quaisquer bytes recebidos, e não transmite. Esse modo é permanente: para voltar à comunicação normal, deve ser manualmente reiniciado. A Tabela 4 mostra os comandos compreendidos pelo módulo de aquisição. Todos os comandos consistem de um único byte, exceto os de atribuições de elemento e grupo, que possuem carga útil. Os nomes indicados na primeira coluna foram arbitrados para melhor legibilidade do código-fonte.

Todas as respostas consistem de uma sequência de bytes, na

Tabela 4 – Comandos compreendidos pelo módulo de aquisição.

Comando	Byte	Descrição
PING	20h	Checagem de comunicação. MA deve responder com uma mensagem PONG
SET_ELEM	21h	Atribuição de nome de elemento. Segue uma quantidade variável de bytes contendo o novo nome de elemento, terminada por quebra de linha (bytes 0Ah ou 0Dh)
SET_GRP	22h	Atribuição de nome de grupo. Segue uma quantidade variável de bytes contendo o novo nome de grupo, terminada por quebra de linha (bytes 0Ah ou 0Dh)
MT_NA	30h	MT não disponível. Sinaliza ao MA para desistir da comunicação, e mostrar ao usuário os avisos convenientes
REQ_ID	31h	Solicitação de identificação. MA deve responder enviando o tipo de dispositivo
REQ_ELEM	32h	Solicitação de nome de elemento. MA deve responder enviando seu nome de elemento
REQ_EGRP	33h	Solicitação de nomes de elemento e grupo. MA deve enviar seus nomes de elemento e grupo
RST_WDTF	34h	Limpar marcação do WDT. MA não deve mais enviar em suas mensagens a indicação de que houve reinicialização por WDT
REQ_RST	35h	Solicitação de reinicialização (*reset*). MA deve forçar seu *hardware* a reinicializar
START	41h	Começar a transmitir dados. MA deve sair do modo de configuração e entrar em modo de transmissão de dados
PAUSE_BO	42h	Pausar a transmissão por *buffer* cheio. MA deve suspender os dados, sem mudar de modo
STOP_CFG	43h	Voltar ao modo de configuração
DATA_ACK	44h	Confirmação de recebimento de dados, enviado após cada dado recebido corretamente

qual o bit mais significativo do primeiro byte é sempre 1, e dos bytes seguintes é sempre zero. Dessa forma, se houver perda de bytes na transmissão, o módulo de transmissão ainda será capaz de identificar o início de uma resposta.

A Figura 13 mostra a estrutura das respostas. O primeiro byte contém três bits que identificam o propósito da mensagem, de forma que sua interpretação independe de contexto. Dois bits são reservados para o tipo de informação, usado em algumas mensagens, e dois bits são de dados. Caso a mensagem contenha mais de dois bits de dados, os bits seguintes seguirão em um ou dois bytes adicionais, sempre com os bits de dados alinhados à direita da seqüência completa. As únicas exceções a essa estrutura são as mensagens de confirmação, explicadas posteriormente.

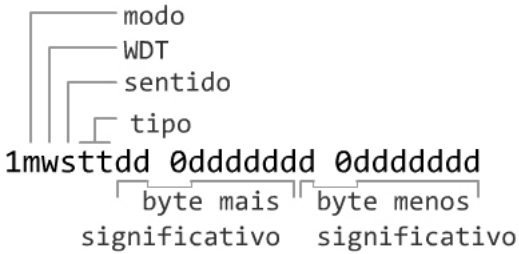

Figura 13 – Estrutura do primeiro byte de uma resposta do módulo de aquisição para o módulo de transmissão. Os significados dos bits representados pelas letras são mostrados na Tabela 5.

A Tabela 5 mostra o significado de cada bit mostrado na Fig. 13, enquanto a Tabela 6 mostra as combinações possíveis entre eles, com os respectivos significados.

A combinação *m=1* e *s=1* (isto é, uma mensagem de transmissão de dados, em que a informação é solicitada ao invés de enviada) não pode ocorrer. Assim, essa combinação foi utilizada para mensagens de confirmação, consistindo em exceções à estrutura. As linhas onde ocorre essa combinação estão marcadas na Tabela 6 com fundo escuro e texto em itálico.

Quando *m=0* (isto é, uma mensagem de configuração), os dois bits seguintes (indicados como *tt* na Fig. 13) indicam qual o tipo de dado de identificação que está sendo transmitido, de acordo com a Tabela 7.

As mensagens de Notificação de Existência (NE) consistem em

Tabela 5 – Significado dos bits da Fig. 13.

Bit	Significado
m	Indica mensagem de configuração (0) ou de dados (1)
w	Indica se o MA foi reinicializado por disparo do WDT, exceto nas mensagens de confirmação
s	Indica se o MA está enviando (0) ou solicitando (1) informações
t	Indica o tipo de informação, onde aplicável
d	Carga útil (dados) transportada pela mensagem

Tabela 6 – Combinações possíveis entre os bits **m**, **w** e **s**, com respectivos significados.

Bits			Significado da mensagem e suas implicações
m	w	s	
0	0	0	Enviando identificação após inicialização normal
0	0	1	Solicitando instruções após inicialização normal
0	1	0	Enviando identificação após reinicialização por WDT
0	1	1	Solicitando instruções após reinicialização por WDT
1	0	0	Enviando dados após inicialização normal
1	*0*	*1*	*ACK (Confirmação de comando recebido)*
1	1	0	Enviando dados após reinicialização por WDT
1	*1*	*1*	*PONG (Resposta a um comando PING)*

Tabela 7 – Combinações possíveis do par de bits *tt*, com respectivos significados, para mensagens de configuração.

Bits	Significado
00	Notificação de Existência (NE) – avisa ao MT que existe um MA conectado ao canal de comunicações
01	Tipo de Dispositivo (TD) – indica ao MT o tipo de dado que será transmitido (entre estado, seqüência e evento)
10	Identificação Textual (IT) – transmite ao MT os nomes de elemento e grupo deste dispositivo

um único byte, com os dois bits menos significativos, *dd*, sendo zero. As mensagens de Tipo de Dispositivo (TD) consistem em um único byte, sendo os dois bits *dd* o número que indica o tipo de dispositivo em si, conforme Tabela 8. As mensagens de Identificação Textual (IT) possuem número variável de bytes contendo o nome do elemento, de no máximo 32 caracteres, e terminado pelo byte 00h, ou terminado pelo byte 01h e seguido do nome do grupo, de no máximo 32 caracteres, terminado por 00h.

Tabela 8 – Identificadores numéricos dos tipos de dispositivo.

Bits	Significado
00	Transmite informações do tipo *estado*
01	Transmite informações em uma *seqüência*
10	Transmite informações do tipo *evento*
11	Reservado para possível uso futuro

Na situação em que *m=1* e *s=0* (isto é, enviando dados de medida), os dois bits *tt* indicam o tamanho da mensagem, de acordo com a Tabela 9.

Tabela 9 – Indicação, através dos bits *tt*, de comprimento da mensagem de dados.

Bits	Tamanho da mensagem
00	8 bits de dados, segue mais um byte após este
01	16 bits de dados, seguem mais dois bytes após este
1x	Reservado para possível uso futuro

A Figura 14 mostra, como exemplo, as etapas de comunicação para um MA do tipo estado, transmitindo dados de 16 bits, do momento em que é ligado até a transmissão de dados.

3.4.4.2 Comunicação entre segundo e terceiro estágios

A comunicação acontece em barramento, com endereçamento por *software* em cada mensagem. O barramento foi implementado utilizando o *hardware* I^2C dos microcontroladores, mas o protocolo não é, rigorosamente falando, I^2C. O protocolo original é proprietário, e os

MA		MT	
c c c t t d d		CMD	SIGNIFICADO
0 0 1 0 0 0 0	→		NE
	←	REQ_ID	Solicita TD
0 0 0 0 1 0 0	→		TD é *estado* (bits dd = 00)
	←	REQ_EGRP	Solicita IT
0 0 0 1 0 0 0	→		Indica que seguirá IT
(2 a 66 bytes)	→		Dados IT
	←	START	Solicita dados
1 0 0 0 1 d d	→		Inicia pacote de dados
0ddddddd 0ddddddd	→		Restante dos dados

Figura 14 – Mensagens trocadas entre um MA e um MT para a configuração e início de transmissão de dados, para um MA do tipo estado e com dados de 16 bits.

endereços são convencionados. Além disso, esses endereços devem ser fixos, um método incompatível com a proposta deste trabalho. Assim, uma variante foi desenvolvida seguindo preceitos semelhantes.

Duas formas de identificação de dispositivo foram empregadas simultaneamente: o endereçamento via protocolo, semelhante ao protocolo I^2C (isto é, um byte compatível com o *hardware* I^2C dos microcontroladores), e um endereçamento físico através de um sinal elétrico (um pino do microcontrolador), semelhante ao *chip select* do protocolo SPI. Neste trabalho, essa estratégia foi denominada *seleção em cascata*, e o sinal digital físico é designado aqui como *dynamic chip select* – DCS (fazendo alusão ao termo para o sinal do qual ele é variante).

Para facilitar a leitura dos códigos-fonte dos *firmwares*, os endereços foram indicados aqui em bytes completos de oito bits, sendo o endereço alinhado à esquerda e o bit menos significativo sempre zero.

Cada dispositivo possui uma entrada digital (*entrada DCS*) e uma saída digital (*saída DCS*), dedicadas ao processo de seleção em cascata. Duas linhas – SDA e SCL – são ligadas em barramento, de forma idêntica ao protocolo I^2C, enquanto as linhas DCS são ligadas em cascata, isto é, a entrada DCS de um dispositivo é ligada na saída DCS do dispositivo anterior, conforme a Fig. 15.

Após a inicialização (*reset*), o dispositivo escravo (módulo de transmissão) não possui endereço. O endereço EEh (238 decimal) é o endereço padrão dos dispositivos que ainda não receberam seu endereço específico. Nesse estado, o dispositivo é controlado pelo DCS, isto é,

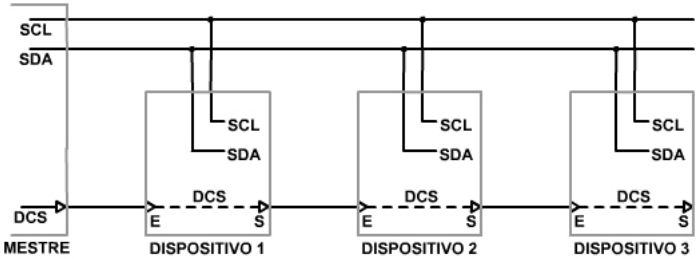

Figura 15 – Diagrama de conexão da seleção em cascata. Linhas sólidas indicam conexão elétrica, enquanto linhas tracejadas indicam conexão por *software*.

responderá se e somente se sua entrada DCS (representada como "E" na Fig. 15) tiver um nível lógico alto. Um nível lógico baixo na entrada DCS desativa o *hardware* I^2C do microcontrolador. Nesse estado (dispositivo não-endereçado), a saída DCS (representada como "S" na Fig. 15) é mantida em zero. Dessa forma, todos os dispositivos que estiverem à frente (na cascata) de um dispositivo sem endereço terão suas entradas DCS em zero, uma vez que estão ligadas à saída DCS anterior.

Dispositivos não-endereçados respondem somente a duas instruções, enviadas ao endereço EEh:

- Uma conversação de leitura: responde com o valor E0h (224 decimal). Essa conversação possui a função de verificar se existe algum dispositivo não-endereçado;
- Uma conversação de escrita: o dispositivo escravo aguarda um – e somente um – byte de dados, que assumirá como seu novo endereço.

Ao receber um novo endereço, o dispositivo abandona o estado não-endereçado, e passa a responder ao endereço recebido ao invés de EEh, independentemente do sinal DCS. A entrada DCS passa a ser replicada na saída DCS, formando uma conexão elétrica virtual por *software*. Nesse estado (endereçado), o dispositivo entra em operação normal, interpretando todos os outros comandos enviados pelo módulo principal.

O mestre busca por dispositivos não-endereçados no barramento a intervalos regulares, através de uma conversação de leitura no en-

dereço EEh com a linha DCS alta. Caso encontre (receba E0h), envia um novo endereço ainda não atribuído, através de uma conversação de escrita. Através dessa estratégia, é possível ter um endereço de barramento atribuído dinamicamente aos dispositivos de forma inequívoca, sequencial e sem a possibilidade de colisão de endereços. Também permite que os dispositivos sejam inseridos no barramento a qualquer momento, sendo endereçados na próxima verificação de dispositivo não-endereçado. A Fig. 16 mostra uma sequência de endereçamento para três dispositivos, onde se pode ver que, a qualquer momento, só existe um dispositivo não-endereçado com a entrada DCS alta.

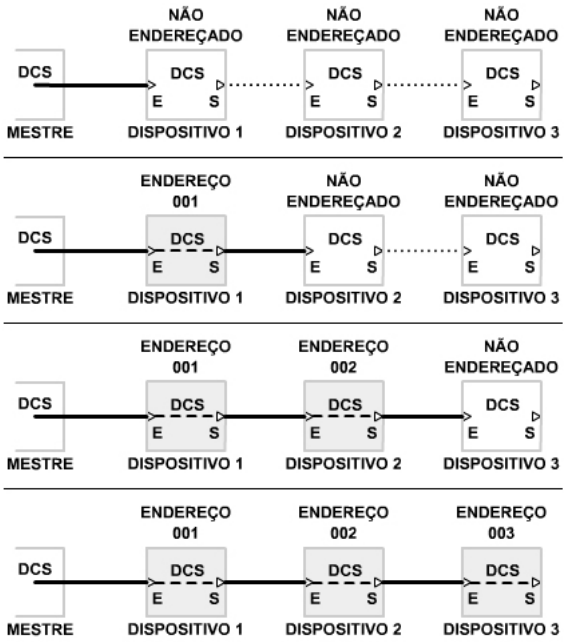

Figura 16 – Passos do endereçamento através de seleção em cascata, para um barramento com três dispositivos escravos. Linha pontilhada indica nível lógico zero (DCS inativo), linha contínua em negrito indica nível lógico alto (DCS ativo). Linha tracejada indica conexão virtual entre entrada e saída DCS.

O endereço usado foi de 7 bits, o que significa um máximo de 128 endereços. Desses, 16 estão reservados pela especificação do protocolo I^2C e não podem ser usados para endereçamento, restando 112

endereços possíveis. O algoritmo implementado suporta até 100 endereços, por razões de legibilidade de código durante o desenvolvimento.

A comunicação entre o segundo e terceiro estágios – uma vez que os dispositivos estejam endereçados – também segue a lógica *comando-resposta*, em que todas as mensagens do módulo principal para o de transmissão são comandos, e as do módulo de transmissão para o principal são respostas. No entanto, como a comunicação é síncrona, o módulo principal pode solicitar várias respostas para um único comando. Nesse caso, para cada resposta solicitada após a primeira, assume-se que o comando seja o mesmo.

Os comandos compreendidos pelo módulo de transmissão estão mostrados na Tabela 10. Numa sessão de escrita, o módulo principal envia o endereço do escravo e o bit de escrita, seguido do byte contendo o comando, e seguido de carga útil onde aplicável.

Tabela 10 – Comandos compreendidos pelo módulo de transmissão. A coluna *Comando* indica o nome do comando, arbitrado neste trabalho; a coluna *Byte* indica o valor do comando em si; a coluna *Descrição* indica o significado do comando.

Comando	Byte	Descrição
REQ_DATA	00h	Requisição de dados. Nas próximas leituras, o MT deve enviar dados que tiver armazenado em *buffer*
REQ_ID	01h	Requisição de identificação. Nas próximas leituras, o MT deve enviar a identificação (ITs e TD) do MA conectado a ele, se houver
PING	20h	Verifica se um MT com este endereço está conectado. Deve responder com um PONG
RST_MA	21h	Solicita que o MT reinicie o MA conectado a ele, isto é, que o MT envie um comando REQ_RST ao respectivo MA
RST_MT	22h	Solicita que o MT reinicie (*reset*)
RENAME	30h	Atribui novo endereço ao MT. O novo endereço deve ser transmitido imediatamente após o comando, e passa a valer a partir do fim da sessão de escrita

Os comandos REQ_DATA e REQ_ID indicam ao módulo de transmissão o tipo de informação que está sendo requisitada. Essa

será a informação enviada em todas as leituras subseqüentes. O comando PING é usado principalmente para verificar se o dispositivo ainda está conectado, ou foi removido pelo usuário – caso tenha sido removido, não deve mais ser lido, sem comprometer o funcionamento do resto do sistema. Em caso de remoção de dispositivo, o comando RENAME é usado para renomear os dispositivos de forma a mantê-los sempre em uma seqüência ininterrupta, para facilitar a varredura. Os comandos RST_MA e RST_MT permitem reiniciar os módulos em caso de comportamento indevido sem perda de comunicação (em caso de travamento com perda de comunicação, apenas o recurso WDT dos respectivos microcontroladores é que pode reiniciar os módulos).

Todas as respostas do módulo de transmissão começam com um byte de cabeçalho. Se o bit mais significativo desse byte for zero (isto é, o byte for menor que 80h), significa uma mensagem de dados (resposta a REQ_DATA), e os 7 bits menos significativos (isto é, o valor do byte em si) indicam a quantidade de dados a serem transmitidos, em valores de 16 bits (dois bytes). Se o byte de cabeçalho for igual a 81h, trata-se de uma resposta ao comando REQ_ID, e seguem-se mais 65 bytes: 32 bytes contendo o nome do elemento, 32 bytes contendo o nome do grupo, e 1 byte contendo o tipo de dispositivo. Se o byte de cabeçalho for E0h, trata-se de um PONG, isto é, uma resposta ao comando PING, e segue-se mais um byte, indicando a situação da conexão com o módulo de aquisição, conforme Tabela 11.

3.4.5 Módulos de Aquisição

A Figura 17 mostra o diagrama de um MA, em uma representação independente da natureza da medida. A cada grandeza física corresponderá uma forma de implementação do sinal indicado na figura como IO (*input/output*). Protótipos foram implementados usando o microcontrolador PIC16F628A.

Os processadores da família 16F não possuem uma instrução interna para reinicialização. Dessa forma, um mecanismo externo foi desenvolvido, através dos pinos $\overline{\text{MCLR}}$ e RA4.

A saída RA4 é construída, fisicamente, na configuração *dreno aberto*, aterrando o sinal quando o seu valor lógico é baixo, mas mantendo a saída em alta impedância quando seu valor lógico é alto. O estado da saída RA4 ao ligar o dispositivo é em alta impedância.

Um nível lógico baixo durante pelo menos 2 μs na entrada $\overline{\text{MCLR}}$ reinicializa o microcontrolador. Esse nível lógico é dado pela tensão do

Tabela 11 – Último byte da resposta a um comando PING, indicando a situação da conexão com um MA.

Situação	Byte	Significado
PONG_MAOFF	00h	Não existe MA conectado
PONG_BLANK	01h	Conectado a um MA, mas as ITs ainda não foram obtidas
PONG_MACFG	02h	Conectado a um MA que está sendo configurado (transmissão de dados de medida não iniciada, ou interrompida)
PONG_MAOK	03h	Conectado a um MA que está transmitindo dados de medida
PONG_MALB	04h	Conectado a um MA que está transmitindo dados de medida, e que está notificando bateria com pouca carga
PONG_LOST	05h	Esteve conectado a um MA, mas a conexão foi perdida

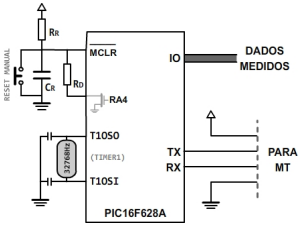

Figura 17 – Diagrama esquemático de um MA, com representação genérica da leitura da medida (IO).

capacitor C_R, que é carregado pelo resistor R_R quando o circuito é alimentado, atingindo e mantendo o nível lógico alto.

Para reinicializar o dispositivo, um nível lógico baixo é colocado na saída RA4, colocando o resistor de descarga R_D em paralelo com o capacitor. Sendo R_D muito menor que R_R, o capacitor é descarregado e um nível baixo é colocado na entrada $\overline{\text{MCLR}}$ até que o dispositivo reinicialize.

O TIMER1 é implementado com um cristal de 32,768 kHz, nos casos em que a medida possui relação com o tempo (como a freqüência cardíaca). Em casos que independem de uma base de tempo (como a temperatura corporal), o temporizador não é necessário.

Para medidas de freqüência, a entrada representada na Fig. 17 como IO consiste em uma entrada digital, recebendo uma transição de nível baixo para alto (rampa de subida) no momento em que ocorra um início de período (como um batimento cardíaco). A partir do segundo pulso, o MA obtém do TIMER1 a diferença de tempo entre este pulso e o anterior (isto é, o período), e calcula a freqüência equivalente, enviando imediatamente uma mensagem de dados ao MT. Assim, a freqüência medida será a freqüência com que as mensagens de dados serão enviadas ao MT, e o tempo entre pulsos será o tempo de validade da informação.

Para medidas de intensidade (como temperatura e pressão), a entrada IO deve ser implementada de acordo com o dispositivo que fornecerá a informação (seja ele um conversor analógico-digital, transdutor com saída digital, ou equipamento externo). Embora não tenha sido implementado, o uso de outro modelo de microcontrolador que possua recursos de leitura de grandezas físicas (como conversor analógico-digital integrado) é possível e compatível com a arquitetura. Como as medidas de intensidade não possuem relação com o tempo, uma freqüência de amostragem deve ser arbitrada e implementada em *firmware*, sendo essa a freqüência de transmissão de dados, e determinando o período de validade da informação.

Para informações do tipo *evento*, a entrada IO consiste em uma entrada digital, que recebe uma rampa de subida no momento da ocorrência de um evento. A freqüência de ocorrência do evento será a freqüência de transmissão de mensagens, e o intervalo entre eventos será o período de validade da informação.

A comunicação com o MT é feita pelo *hardware* USART integrado. Quatro linhas conectam o MA ao MT: transmissão e recepção UART, alimentação e terra. No caso de conexão por fio, alimentação e terra são recebidos do MT. No caso de conexão sem fio, o circuito

é alimentado por baterias, e alimentação e terra são fornecidos para o rádio do MT.

As informações de elemento e grupo são armazenadas em memória EEPROM, e podem ser configuradas através do protocolo (conforme descrito), ou uma interface pode ser desenvolvida no próprio equipamento. Para configuração por protocolo, um dispositivo pode ser implementado atuando como configurador: ao invés de ser conectado a um MT, o MA é conectado ao configurador, que atualiza as ITs, e em seguida o MA é novamente conectado a um MT.

3.4.6 Módulos de Transmissão

A Figura 18 mostra o diagrama de um MT, com conexão ao respectivo MA por rádio. O rádio não foi especificado, visto que a escolha é transparente ao MT – assim como a escolha de não usar rádio. Protótipos foram implementados usando o microcontrolador PIC16F73, com ligação por fio, Bluetooth e XBee.

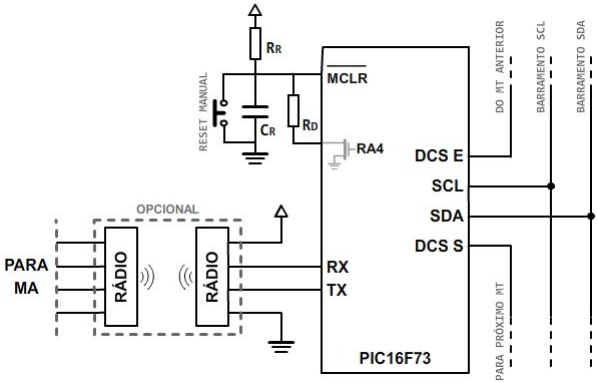

Figura 18 – Diagrama esquemático de um MT.

O mecanismo de reinicialização é o mesmo explicado para o MA, na Seção 3.4.5.

Para a comunicação com o MA (intermediada ou não por rádios), foi usado o *hardware* USART integrado. Quatro linhas são fornecidas: transmissão e recepção UART, alimentação e terra. No caso de conexão por fio, a alimentação e terra são fornecidas diretamente para o MA, enquanto no caso de comunicação por rádio, são fornecidas para o rádio

da ponta conectada ao MT – a alimentação do rádio da ponta conectada ao MA é fornecida pelo MA, por baterias.

Para a comunicação com o módulo principal, via barramento I^2C, foi utilizado o *hardware* integrado SPI/I^2C. Para as linhas (entrada e saída) DCS, foram usados dois pinos digitais da porta C (RC2 para saída e RC5 para entrada). Foi implementado um *buffer* de 32 amostras de 16 bits.

3.4.7 RTC

A Figura 19 mostra o diagrama do *Real Time Clock* (RTC), usando um PIC 16F73.

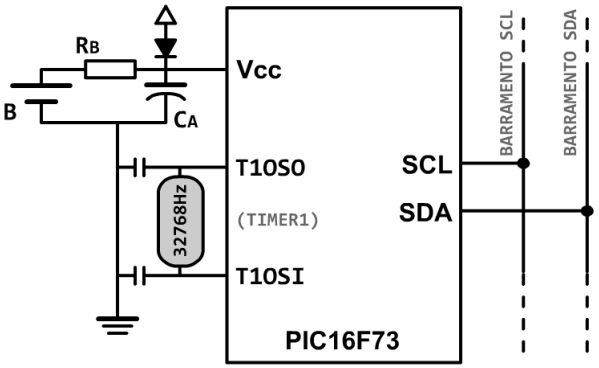

Figura 19 – Diagrama esquemático do RTC.

A alimentação do RTC é feita simultaneamente por duas fontes. Em situação normal, é alimentado diretamente pelo módulo principal. No entanto, como sua função é fornecer contagem de tempo, deve ser mantido alimentado de forma idealmente ininterrupta. Assim, uma bateria B foi implementada em paralelo. No protótipo de testes, o circuito de carga e descarga da bateria consiste de apenas um resistor R_B, e um capacitor C_A é usado em paralelo com ambas as alimentações, para filtrar quaisquer flutuações e permitir um funcionamento estável do microcontrolador. Foi também usado um diodo para manter a alimentação da bateria confinada ao RTC.

O TIMER1 do microcontrolador foi usado, com um cristal de 32,768 kHz, para a contagem de tempo.

Para a comunicação com o módulo principal, via barramento

I^2C, foi utilizado o *hardware* integrado SPI/I^2C, da mesma forma que o modulo de transmissão. São lidos 6 bytes, para ano, mês, dia, hora, minuto e segundo. O endereço foi fixado em ECh/EDh (236 e 237 decimais), por estar fora da faixa usada pelos módulos de transmissão.

3.4.8 Módulo Principal

O módulo principal foi implementado usando uma placa comercial, modelo XBEENET, fornecida pelo fabricante Microgenios, já incluindo conector para Ethernet e componentes necessários para o funcionamento do microcontrolador PIC18F87J60.

3.4.9 Servidor *Web*

A página *web* exibida pelo módulo principal foi desenvolvida visando o monitoramento simultâneo, numa mesma tela, de todos os dispositivos conectados, e atualizando-se automaticamente caso algum novo dispositivo seja conectado, ou algum dos atuais seja removido. A idéia central é que o usuário nunca precise interagir com a página, mesmo se houver alteração na rede de sensores.

O sistema *web* foi estruturado em uma página dinâmica (por AJAX) e duas folhas de dados, totalizando quatro arquivos:

- *index.html* - contém o esqueleto principal e chama as funções JavaScript. Parte dessas funções está no próprio arquivo, parte em uma biblioteca externa;
- *mchp.js* - biblioteca externa de funções JavaScript, fornecida pelo fabricante do microcontrolador, que implementa o processo de requisição AJAX;
- *eleminfo.xml* - folha de dados que organiza as informações de um determinado MA;
- *status.xml* - folha de dados que organiza os valores lidos e as informações sobre o MP.

A página é exibida inicialmente sem quaisquer dados, recebendo apenas o número de dispositivos atualmente conectados. Possui duas rotinas de chamadas AJAX:

- Exibir dispositivos conectados: consulta o arquivo eleminfo.xml para cada dispositivo conectado (passando o número do dispositivo como variável de requisição GET), recebendo suas descrições textuais, tipo de dispositivo e valor atual;

- Atualizar valores: consulta o arquivo status.xml, repetidamente, recebendo (em cada um dos acessos) os valores numéricos de todos os dispositivos, o número de dispositivos conectados, data e hora do acesso, e um valor que indica se houve alterações na rede desde o último acesso.

Assim que carregada, a página executa a rotina para exibir os dispositivos conectados. Para cada requisição, constrói uma linha, na página, para exibição do valor. Caso já exista um outro dispositivo com a mesma descrição de grupo, a linha é inserida na caixa desse grupo; do contrário, uma nova caixa é criada, contendo a descrição do grupo, e a nova linha. Em ambos os casos, a linha contém a descrição do elemento e o valor.

3.5 VERIFICAÇÃO

No âmbito deste trabalho, foi considerado *verificação* como sendo a correspondência entre o valor numérico já digitalizado, apresentado a um módulo de aquisição, e o valor representado em forma humanamente legível, pelo módulo principal, visto em um navegador *web*. Essa verificação desconsidera taxas de atualização da informação, visto que depende de fatores externos, como a infra-estrutura de rede do local. Refere-se à verificação dos protocolos desenvolvidos e da topologia em que se organizam, e portanto a obtenção das medidas, a partir de seus respectivos meios físicos, não deve ser incluída.

Dois processos de verificação foram realizados, para os dois tipos de dados compreendidos neste trabalho: um para o algoritmo de estado, e um para o de evento. O resultado pode ser extrapolado para todos os dados que compartilham o mesmo algoritmo (isto é, verificar dois sinais de estado, como a freqüência cardíaca e a temperatura, seria uma redundância. Um único já verifica todos os sinais de estado, independentemente de suas interpretações físicas).

3.5.1 Verificação do Algoritmo de Estado

Para o procedimento de verificação, foi escolhida como sinal a freqüência cardíaca, por dois motivos: sendo representada por um pulso digital, a leitura é imune a interferências de quaisquer ruídos, na bancada, que possam causar pequenas oscilações de amplitude nos meios de condução; e também por não necessitar de um processo de conversão analógico-digital, simplificando o processo e evitando erros que possam advir do processo de conversão. A única fonte física de erros é o cristal oscilador do microcontrolador.

A verificação consistiu em apresentar, ao módulo de aquisição, pulsos gerados pelo gerador de funções, em um conjunto de valores ao longo da faixa típica de freqüências cardíacas. A menor freqüência considerada foi a que consta no *Guinness Book* como a menor já registrada até o ano de 2005, de 27 bpm, correspondendo a 0,45 Hz. A maior freqüência considerada foi a maior possível pela equação de *Karvonen*, isto é, 220 bpm, correspondendo a 3,67 Hz. Também foram testados os valores intermediários de 60 bpm (1 Hz), 120 bpm (2 Hz) e 180 bpm (3 Hz).

3.5.2 Verificação do Algoritmo de Evento

Para a verificação de eventos, foi utilizado um botão, pressionado em momentos aleatórios. As informações comparadas foram a hora e data observadas na página web no momento em que o botão foi pressionado, e o valor registrado pelo sistema como hora e data do evento.

3.6 AVALIAÇÃO

Neste trabalho, considerou-se como *avaliação* as medidas de desempenho e limites de trabalho, e suas respectivas implicações. Parte desses limites foi imposta por decisões arbitradas na metodologia, enquanto outra parte é conseqüência do carregamento do sistema. No capítulo 4, são apresentadas as limitações decididas e medidas relevantes para análise de carregamento, enquanto no capítulo 5 serão abordadas as respectivas implicações.

4 RESULTADOS

4.1 MEDIDAS DE TEMPO DO LAÇO PRINCIPAL, SEM MÓDULOS

A duração do laço principal do módulo principal após a implementação da Ethernet, e antes da implementação da comunicação com os módulos de transmissão e aquisição, foi medida através de um pulso conforme descrito na Seção 3.3.2

A Figura 20 mostra o intervalo entre pulsos, na coluna da esquerda por persistência, e na coluna da direita por média, sendo a linha de cima sem requisições, e a linha de baixo com requisições AJAX a cada 100ms. Para os casos de persistência o tempo de captura foi de aproximadamente 5 minutos, e de média foi de 64 amostras.

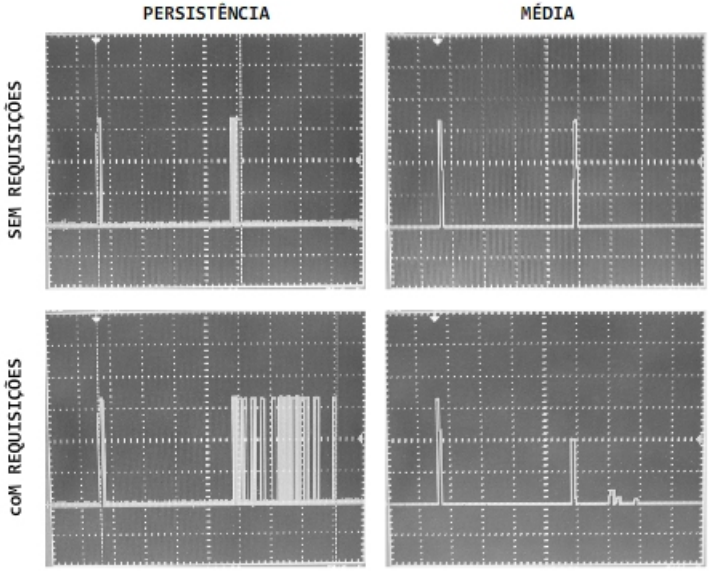

Figura 20 – Duração do laço principal, medido por persistência e por média, sem e com requisições AJAX a cada 100ms.

A duração máxima do laço sem requisições foi aproximadamente 4,5ms, com pequeno espalhamento. A duração máxima do laço principal com requisições foi de aproximadamente 7,5ms, com um espalha-

mento bem maior.

4.2 VERIFICAÇÃO

4.2.1 Verificação do Algoritmo de Estado

As Tabelas 12, 13 e 14 mostram os valores medidos utilizando respectivamente módulo de transmissão por fios, *Bluetooth* e XBee

Tabela 12 – Freqüências usadas para verificação, com conexão por fios, e os respectivos valores mostrados pelo equipamento.

Freqüência no gerador de funções (Hz)	Freqüência por minuto correspondente (bpm)	Valor mostrado pelo equipamento
0,455 ± 0,005	27,30	27
1,000 ± 0,005	60,00	60
2,000 ± 0,002	120,00	120
3,003 ± 0,001	180,18	180
3,676 ± 0,001	220,44	220

Tabela 13 – Freqüências usadas para verificação, com conexão por Bluetooth, e os respectivos valores mostrados pelo equipamento.

Freqüência no gerador de funções (Hz)	Freqüência por minuto correspondente (bpm)	Valor mostrado pelo equipamento
0,458 ± 0,005	27,48	27
1,000 ± 0,005	60,00	60
2,000 ± 0,002	120,00	120
3,007 ± 0,001	180,42	180
3,653 ± 0,001	219,18	219

Tabela 14 – Freqüências usadas para verificação, com conexão por XBee, e os respectivos valores mostrados pelo equipamento.

Freqüência no gerador de funções (Hz)	Freqüência por minuto correspondente (bpm)	Valor mostrado pelo equipamento
0,458 ± 0,005	27,48	27
1,000 ± 0,005	60,00	60
2,000 ± 0,002	120,00	120
3,008 ± 0,001	180,48	180
3,674 ± 0,001	220,44	220

4.2.2 Verificação do Algoritmo de Evento

Dez medidas foram realizadas, por fios, bluetooth e XBee, constando respectivamente nas Tabelas 15, 16 e 17.

Tabela 15 – Hora e data observados na página web no momento do evento, e respectivos valores registrados pelo sistema como hora e data do evento, transmitido por fios.

Medida	Hora e Data Observadas	Hora e Data Registradas
1	23:10:40 - 04/01/2013	23:10:40 - 04/01/2013
2	23:11:08 - 04/01/2013	23:11:08 - 04/01/2013
3	23:11:25 - 04/01/2013	23:11:26 - 04/01/2013
4	23:13:25 - 04/01/2013	23:13:25 - 04/01/2013
5	23:13:42 - 04/01/2013	23:13:42 - 04/01/2013
6	23:14:02 - 04/01/2013	23:14:02 - 04/01/2013
7	23:14:30 - 04/01/2013	23:14:31 - 04/01/2013
8	23:14:51 - 04/01/2013	23:14:51 - 04/01/2013
9	23:15:04 - 04/01/2013	23:15:05 - 04/01/2013
10	23:15:25 - 04/01/2013	23:15:25 - 04/01/2013

Tabela 16 – Hora e data observados na página web no momento do evento, e respectivos valores registrados pelo sistema como hora e data do evento, transmitido por *Bluetooth*.

Medida	Hora e Data Observadas	Hora e Data Registradas
1	22:49:05 - 04/01/2013	22:49:05 - 04/01/2013
2	22:52:50 - 04/01/2013	22:52:50 - 04/01/2013
3	22:53:13 - 04/01/2013	22:53:13 - 04/01/2013
4	22:53:32 - 04/01/2013	22:53:33 - 04/01/2013
5	22:55:02 - 04/01/2013	22:55:02 - 04/01/2013
6	22:55:18 - 04/01/2013	22:55:18 - 04/01/2013
7	22:55:41 - 04/01/2013	22:55:42 - 04/01/2013
8	22:56:04 - 04/01/2013	22:56:04 - 04/01/2013
9	22:56:39 - 04/01/2013	22:56:39 - 04/01/2013
10	22:56:53 - 04/01/2013	22:56:54 - 04/01/2013

Tabela 17 – Hora e data observados na página web no momento do evento, e respectivos valores registrados pelo sistema como hora e data do evento, transmitido por XBee.

Medida	Hora e Data Observadas	Hora e Data Registradas
1	01:45:20 07/01/2013	01:45:20 07/01/2013
2	01:46:25 07/01/2013	01:46:25 07/01/2013
3	01:46:41 07/01/2013	01:46:41 07/01/2013
4	01:46:57 07/01/2013	01:46:57 07/01/2013
5	01:47:13 07/01/2013	01:47:14 07/01/2013
6	01:47:39 07/01/2013	01:47:39 07/01/2013
7	01:47:52 07/01/2013	01:47:52 07/01/2013
8	01:48:08 07/01/2013	01:48:08 07/01/2013
9	01:48:27 07/01/2013	01:48:27 07/01/2013
10	01:48:48 07/01/2013	01:48:48 07/01/2013

4.3 AVALIAÇÃO

4.3.1 Comunicação entre Primeiro e Segundo Estágios

Para um canal UART (entre MA e MT) de 19200 bps, cada palavra de dados consistindo em 8 bits úteis, 1 *start bit* e 1 *stop bit*, temos 1920 bytes úteis por segundo. Cada mensagem de dados (para amostras de 16 bits) do MA para o MT utiliza 3 bytes, logo temos um máximo de 640 amostras por segundo do MA para o MT.

4.3.2 Comunicação entre Segundo e Terceiro Estágios

Os tempos de transmissão de dados no barramento I^2C, medidos na linha SCL com somente o RTC (isto é, nenhum MT), são mostrados na Tabela 18. Três medidas foram feitas, sucessivamente.

Tabela 18 – Medidas de tempo total, de transmissão e entre transmissões para a linha SCL, tendo somente o RTC conectado.

Medida	Tempo total (ms)	Tempo de transmissão (ms)	Tempo entre transmissões (ms)
1	5,88 ± 0,01	1,64 ± 0,01	4,24 ± 0,01
2	5,88 ± 0,01	1,64 ± 0,01	4,24 ± 0,01
3	5,88 ± 0,01	1,64 ± 0,01	4,24 ± 0,01

O tempo entre dois inícios sucessivos de comunicação, medido no RTC (isto é, o tempo total), foi medido através da alternância de uma saída lógica a cada início de sessão I^2C. Medidas foram feitas sucessivamente adicionando MTs, de nenhum MT até 3 MTs. Os MTs estavam todos conectados a MAs que não transmitiam dado algum. O tempo na situação de nenhum MT conectado – isto é, lendo somente o RTC – foi medido por média para obtenção do valor típico (indicado pelo osciloscópio como 5,874 ms), e por persistência para identificação dos limites, ambos capturados durante 2 minutos. Estão representados, respectivamente, nas figuras 21.a e 21.b. As medidas limítrofes de tempo, por persistência de 2 minutos, estão na Tabela 19.

A diferença dos tempos entre uma situação da Tabela 19 e a situação com o número anterior de MTs (isto é, o tempo adicional

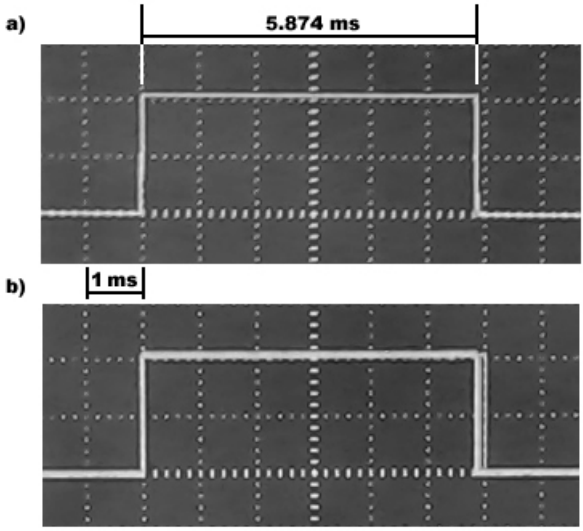

Figura 21 – Medidas de tempo entre duas leituras sucessivas do RTC, sem MTs conectados, sendo a) média; b) persistência.

Tabela 19 – Medidas de tempo mínimo e máximo entre leituras do RTC, para vários valores de MTs conectados.

MTs	Tempo mínimo (ms)	Tempo máximo (ms)
0	5,84 ± 0,01	6,04 ± 0,01
1	7,32 ± 0,01	7,54 ± 0,01
2	8,80 ± 0,01	9,02 ± 0,01
3	10,28 ± 0,01	10,50 ± 0,01

por MT, calculado subtraindo-se o valor pelo valor da linha acima) é mostrada na Tabela 20.

Tabela 20 – Diferença de tempo de laço principal entre uma determinada situação, e a situação com um MT a menos, isto é, o tempo de processamento e comunicação adicionado por cada MT.

MTs	Tempo mínimo (ms)	Tempo Máximo (ms)
1	1,48 ± 0,02	1,50 ± 0,02
2	1,48 ± 0,02	1,48 ± 0,02
3	1,48 ± 0,02	1,48 ± 0,02

As atividades de leitura de dispositivos nos barramentos I^2C, nas situações mostradas nas Tabelas 18 e 19 estão representadas na figura 22. As situações **a**, **b**, **c** e **d** são, respectivamente, 0, 1, 2 e 3 MTs conectados. O sinal representado é o envelope do sinal real, isto é, um nível alto indica comunicação ocorrendo (leitura), enquanto um nível baixo indica silêncio nos barramentos – e processamento do módulo principal. As linhas verticais em negrito indicam o início da rotina de leitura.

O tempo entre dois inícios sucessivos de transmissão, no RTC, com dados sendo lidos em um MT (isto é, o respectivo MA não estava em silêncio), foi medido para dados de 16 bits em duas situações: com dados sintéticos de vários tamanhos, e com dados reais em várias freqüências.

Na primeira situação, dados sintéticos de comprimento fixo foram transmitidos de forma idêntica em todas as sessões de leitura, para medida (por média e persistência, 2 minutos) de tempo gasto por tamanho de mensagem, em 5 quantidades de dados diferentes (de 0 a 3 amostras, e 10 amostras). A Tabela 21 mostra o tempo médio para cada caso (isto é, o tempo diretamente medido), e o tempo médio da última amostra, isto é, a diferença entre o valor e o anterior. Para a linha de 10 amostras, a subtração foi feita em relação ao valor de 1 amostra, e esse valor foi assumido como uniformemente distribuído pelas 9 amostras do intervalo da diferença. As imagens de média e persistência são mostradas na Fig. 23.

Na segunda situação, dados reais foram transmitidos em 5 freqüências diferentes, por persistência (2 minutos) e média (2 minutos) respectivamente conforme Tabela 22 e Fig. 24.

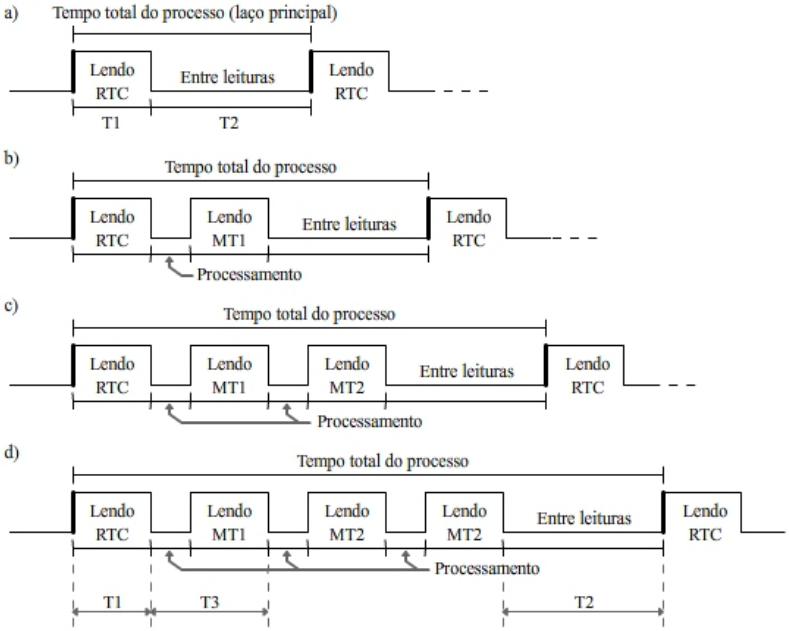

Figura 22 – Representação da atividade de leitura nos barramentos I^2C, nas situações a) nenhum MT conectado; b) um MT conectado; c) dois MTs conectados; d) três MTs conectados.

Tabela 21 – Tempo médio entre transmissões para um MT transmitindo quantidade conhecida de amostras em todas as leituras, e tempo médio acrescido por amostra, pelo aumento de amostras em relação à linha anterior.

Nº de amostras	Tempo médio (ms)	Tempo médio da última amostra (ms)
0	7,348 ± 0,004	—
1	7,544 ± 0,004	0,196 ± 0,008
2	7,864 ± 0,004	0,320 ± 0,008
3	8,184 ± 0,004	0,320 ± 0,008
10	10,44 ± 0,01	0,322 ± 0,002

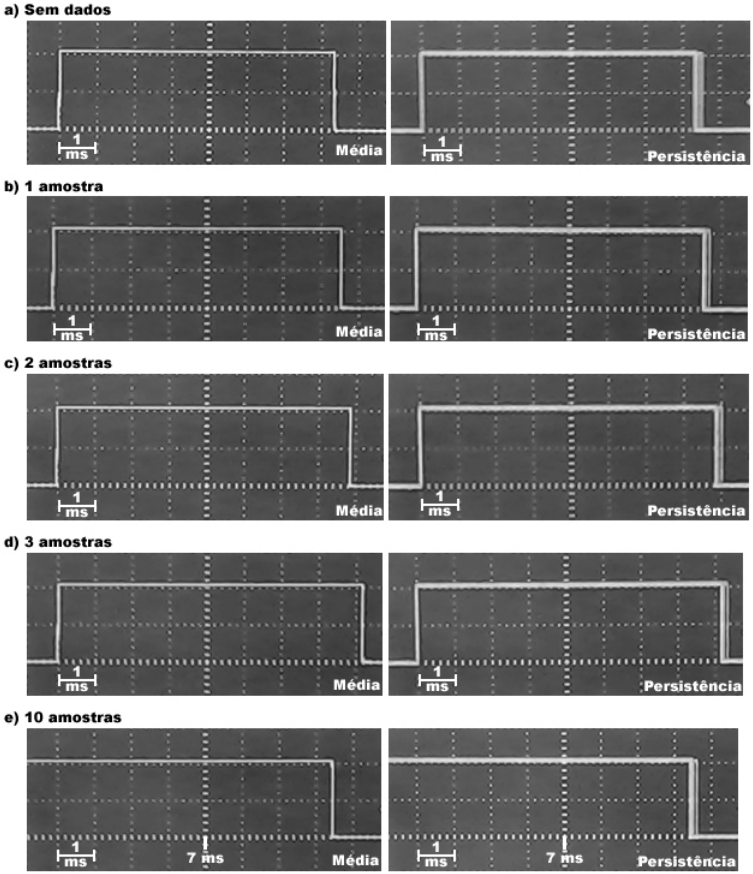

Figura 23 – Medidas de tempo entre leituras do RTC, com um MT conectado transmitindo dados de 16 bits em todas as sessões de leitura, para 5 quantidades diferentes de amostras.

Tabela 22 – Medidas de tempo mínimo e máximo entre leituras do RTC, com um MT conectado transmitindo dados de 16 bits, para 5 freqüências diferentes de transmissão de dados.

Freqüência	Tempo mínimo (ms)	Tempo máximo (ms)
3.0 ± 0,1 Hz	7,33 ± 0,01	7,60 ± 0,01
10,0 ± 0,1 Hz	7,33 ± 0,01	7,64 ± 0,01
100 ± 1 Hz	7,33 ± 0,01	7,71 ± 0,01
200 ± 1 Hz	7,52 ± 0,01	8,04 ± 0,01
300 ± 1 Hz	7,84 ± 0,01	8,37 ± 0,01

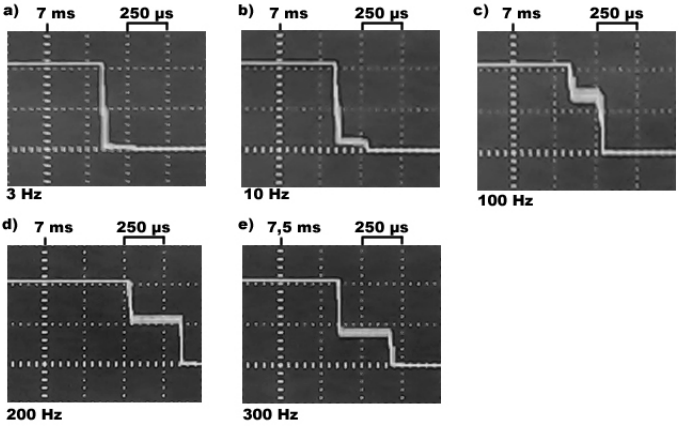

Figura 24 – Medidas de tempo entre leituras do RTC, por média, com um MT conectado transmitindo dados de 16 bits, para 5 freqüências diferentes de transmissão de dados.

5 DISCUSSÃO

5.1 MEDIDAS DE TEMPO DO LAÇO PRINCIPAL, SEM MÓDULOS

Na Fig. 20, na maior parte das execuções com requisições AJAX, o laço teve a mesma duração da situação sem requisições, mostrando esporadicamente alguns laços mais longos.

Esse valor significa uma velocidade de processamento virtualmente paralelo de aproximadamente 133 iterações de laço principal por segundo, no pior caso, e aproximadamente 222 iterações típicas.

Os valores mostrados na Seção 4.1 basearam a decisão de protocolo para os módulos de transmissão, descartando o uso da UART como meio único, e incluindo um barramento I^2C, conforme mencionado na seção 3.4.4.

Assumindo:

- a topologia proposta, em que o módulo principal é mestre da comunicação;
- o melhor caso de carga, isto é, com apenas um módulo de aquisição;
- a mensagem de comando mais curta possível, isto é, um byte;
- a resposta mais curta possível, ou seja, apenas um valor numérico da medida (sem cabeçalhos);
- uma medida composta de 16 bits (2 bytes),

teremos uma comunicação composta de três bytes (um enviado e dois recebidos), e nada mais. A forma de recepção que terá menor impacto no buffer *Ethernet* é ler somente um byte por iteração do laço, o que a 133 iterações por segundo significa 44,3 amostras por segundo. Supondo a maior situação de carga permitida por um endereçamento de um byte (255 módulos de aquisição), e mantendo o resto do cenário, temos aproximadamente 0,174 amostras por segundo, ou uma amostra a cada 5,75 segundos.

Embora esse valor esteja suficiente para alguns casos, como por exemplo, níveis de glicose em diabéticos, é insuficiente para outros, como por exemplo freqüência cardíaca – atualizada a cada batimento. Essa taxa de amostragem também impediria o acompanhamento de formas de onda como o eletrocardiograma.

Além de inaptos, esses valores não estão sequer na mesma ordem de grandeza desejada. Dessa forma, a recepção feita por UART pelo módulo principal foi considerada inviável, decidindo-se por um *buffer* intermediário no módulo de transmissão, descarregado por I^2C.

5.2 VERIFICAÇÃO

5.2.1 Verificação do Algoritmo de Estado

Analisando individualmente cada uma das Tabelas 12, 13 e 14, observamos a verificação do algoritmo. Ao mesmo tempo, observando a invariabilidade entre as três tabelas, verificamos a independência do sistema em relação à tecnologia de transmissão utilizada pelo MT.

Nenhum dos valores de freqüência utilizados resultou numa freqüência cardíaca correspondente com valor decimal acima de 0,5. Assim, não podemos supor que o arredondamento dos valores esteja de acordo com quaisquer padrões estipulados. A faixa de arredondamento pode ser inferida pela análise do código do firmware, mas no âmbito desta verificação, será considerada como desconhecida. O máximo desvio de um valor inteiro foi o valor 180,48, representado na página web como 180. Assim, há certeza de que uma diferença entre 0 e 0,48 será arredondada para baixo, implicanto que o ponto de arredondamento está entre 0,52 e 0,99. Dentro dessa faixa possível, será assumido o pior caso, isto é, de uma operação de truncamento, em que o valor decimal é simplesmente descartado (por exemplo, 180,99 será exibido como 180).

Dessa forma, está verificado que o sistema mostrará a informação com um erro máximo não menor que 1,0 – dentro da faixa de freqüência considerada.

5.2.2 Verificação do Algoritmo de Evento

O processo de verificação forneceu dados relevantes, mas possui dois vieses graves em relação à precisão: o fator humano na observação dos horários exatos, e os atrasos na exibição *web* causados pela infraestrutura de rede.

Estipulou-se três fontes de divergência nas medidas: primeiro os atrasos na atualização do relógio pela rede. Em seguida, o atraso do observador em pressionar o botão. Por fim, a captura do evento e correspondente exibição pelo sistema. Nenhuma dessas fontes é men-

surável individualmente, na forma como a verificação foi realizada. Assim, deve-se considerar que as divergências das tabelas 15, 16 e 17 são referentes à soma de todas as fontes. Essa soma atingiu o valor máximo de 1 segundo, que é portanto a precisão verificada do sistema.

Assim como ocorreu com o algoritmo de estado, também para o algoritmo de evento a semelhança entre as três tabelas demonstra a independência do sistema à tecnologia de transmissão do MT.

5.3 AVALIAÇÃO

Na Figura 22.a, temos a situação da Tabela 18. O tempo T1 é o tempo de transmissão (1,64 ms), e o tempo T2 é o tempo entre transmissões (4,24 ms). Os valores entre início de transmissão, mostrados na Tabela 19 e Fig. 21, referem-se à soma T1+T2. O tempo T1 é exclusivamente ocupado por um processo determinístico e de tamanho fixo (comunicação I^2C), enquanto T2 varia de acordo com a tarefa de rede, logo a variação de duração da Tabela 19 deve-se a T2.

A cada MT adicionado, soma-se ao tempo total um tempo de processamento e o tempo de leitura do MT propriamente dito. A soma do tempo de processamento e do tempo de leitura, mostrada como T3 na Figura 22, é a diferença mostrada na Tabela 20 (tipicamente 1,48 ms). Esse valor é para um MT cujo MA não está transmitindo dados.

Dessa forma, o tempo total do processo será dado pela equação:

$$T_{Total} = T1 + T3 \times N_{MTs} + T2$$

Onde T_{Total} é o tempo total entre dois inícios de transmissão, e N_{MTs} é a quantidade de MTs conectados. Para um tempo $T1 + T2$ típico de 5,88ms, e um $T3$ típico de 1,48ms, podemos escrever a equação como:

$$T_{Total} = 5,88 + 1,48 \times N_{MTs}$$

Onde T_{Total} será dado em milissegundos. Os valores da equação para N_{MTs} de 0 a 3 estão dentro da faixa da Tabela 19, confirmando a equação.

O tempo gasto pela transmissão de um MT será dado pela equação:

$$T_{Total} = T_{BaseTotal} + T_{Am} \times N_{Am}$$

Onde T_{Total} é o tempo total entre dois inícios de transmissão, T_{Am} é o tempo médio por amostra da sessão, N_{Am} é o número de

amostras transmitidas na sessão, e $T_{BaseTotal}$ é um tempo hipotético, gasto com o resto da sessão de comunicação, com leitura do RTC e com processamento. Os tempos T1 e T2 estão inclusos em $T_{BaseTotal}$, sendo o restante gasto na sessão de leitura do MT e processamento dos dados. O tempo gasto na leitura (e conseqüente processamento) de um MT, excetuando-se o tempo de transmissão das amostras propriamente ditas, será referido como T_{MT}. Assim, T_{Total} será dado por:

$$T_{Total} = (T1 + T2) + T_{MT} + T_{Am} \times N_{Am}$$

T_{MT} pode ser estimado por:

$$T_{MT} = T_{Total} - (T1 + T2) - T_{Am} \times N_{Am}$$

Assumindo os valores da Tabela 21 como referência aproximada, e $T1+T2$ como 5,88ms, temos os valores de T_{MT} e cada caso na Tabela 23 a partir de 2 amostras. (Os cálculos para 0 e 1 amostras não foram incluídos, uma vez que foram obtidos exatamente pela operação inversa. Aplicá-los nesta tabela apenas retrocederia a equação em que surgiram, não fornecendo nova informação e carecendo de qualquer consistência.)

Tabela 23 – Valores estimados para T_{MT}.

N_{Am}	T_{Total} (ms)	T_{Am} (ms)	T_{MT} (ms)
2	7,864	0,320	1,344
3	8,184	0,320	1,344
10	10,44	0,322	1,34

Considerando um T_{MT} de 1,344ms, os cálculos de tempo total coincidem com os valores da Tabela 21, exceto para o caso de nenhuma amostra transmitida, em que o T_{Total} calculado vale 7,224ms e o tempo gasto foi 7,348ms, ou seja, um adicional de tempo de 0,124ms quando nenhuma amostra é transmitida. A causa desse tempo adicional é irrelevante para os próximos cálculos.

O T_{Total} (em ms) para um determinado número de amostras não nulo é então representado aproximadamente pela equação:

$$T_{Total} = 7,224 + 0,322 \times N_{Am}$$

O inverso dos valores de tempo da Tabela 21 indicam a máxima freqüência de amostragem que pode ser transmitida por uma determinada quantidade de amostras de (16 bits) por sessão de leitura. Essas

freqüências são mostradas na Tabela 24.

Tabela 24 – Freqüência máxima de amostragem apta a ser transmitida por um determinado número de amostras por sessão de leitura, com somente um MT conectado, obtidos de tempos medidos.

Quantidade de Amostras por Leitura	Freqüência Máxima de Amostragem (amostras/s)
1	132,56
2	254,32
3	366,57
10	957,85

A máxima freqüência (em Hz) para um número arbitrário de amostras pode ser aproximadamente representada pela equação:

$$F_{max} = \frac{N_{Am} \times 1000}{7,224 + 0,322 \times N_{Am}}$$

A Tabela 25 mostra os valores de freqüência máxima obtidos por essa equação na faixa de 1 a 10 amostras. Comparando (onde possível) com os valores da Tabela 24, o maior erro foi para 3 amostras, de 0,074%.

Tabela 25 – Freqüência máxima de amostragem apta a ser transmitida por um determinado número de amostras por sessão de leitura, com somente um MT conectado, obtidos por equação modelada.

Quantidade de Amostras por Leitura	Freqüência Máxima de Amostragem (amostras/s)
1	132,52
2	254,19
3	366,30
4	469,92
5	566,00
6	655,30
7	738,55
8	816,33
9	889,15
10	957,49

Na Tabela 22, as três primeiras freqüência (3 Hz, 10 Hz e 100 Hz) estão contidas inteiramente dentro do limiar de 132,56 Hz, ou seja, nenhuma mensagem de dados conterá mais do que uma amostra. Sendo essas freqüências menores do que o limiar, existirão sessões de leitura sem nenhuma amostra a transmitir.

Para a freqüência de 3 Hz, temos 3 mensagens de 1 amostra (7,544 ms), somando 22,632 ms. Os 977,368 ms restantes são ocupados por leituras sem amostras (de 7,348 ms), sendo em média 133,01 mensagens. Ou seja, 2,2% das mensagens conterão uma amostra, durando até 7,544, contra 97,8% de mensagens que terminarão em 7,348. Esse caso é comprovado pela Figura 24.a.

Para 10 Hz, temos 10 mensagens de 1 amostra, somando 75,44 ms. Os 924,56 ms restantes correspondem a uma média de 125,82 mensagens sem amostras. Isso significa 7,36% de mensagens de 1 amostra, contra 92,64% de mensagens sem amostras. Esse caso é comprovado pela Figura 24.b.

Para 100 Hz, temos 100 mensagens de 1 amostra, somando 754,4 ms. Os 245,6 ms restantes correspondem a uma média de 33,42 mensagens sem amostras. Isso significa 74,95% de mensagens de 1 amostra, contra 25,05% de mensagens sem amostras. Esse caso é comprovado pela Figura 24.c.

A freqüência de 200 Hz está acima do limiar de uma amostra, e abaixo do limite para duas amostras. Logo, parte das mensagens conterá uma amostra, parte conterá duas. O total de amostras dentro de um segundo será 200. Se o número de mensagens contendo uma amostra for N_{1Am} e o número de mensagens contendo duas amostras for N_{2Am}, temos:

$$(N_{1Am} \times 1) + (N_{2Am} \times 2) = 200$$

As mensagens contendo uma amostra ocupam 7,544 ms, enquanto as mensagens contendo duas amostras ocupam 7,864 ms. Assim, ao mesmo tempo, também temos (para tempos expressos em ms):

$$(N_{1Am} \times 7,544) + (N_{2Am} \times 7,864) = 1000$$

Resolvendo esse sistema, temos uma média de 59,136 para N_{1Am} e 70,432 para N_{2Am}, o que significa uma razão de 45,64% de mensagens com uma amostra contra 54,36% de mensagens com duas amostras. Esse caso é comprovado pela Figura 24.d.

A freqüência de 300 Hz está acima da capacidade de duas amostras, e abaixo da de três amostras. Assim, parte das mensagens con-

terão duas amostras, parte, três. O cálculo é similar ao para 200 Hz, ou seja, para 300 Hz temos 300 amostras em um segundo, e portanto:

$$(N_{2Am} \times 2) + (N_{3Am} \times 3) = 300$$

As mensagens com duas amostras ocupam 7,864 ms, enquanto as com três amostras ocupam 8,184 ms. Assim, temos (tempos em ms):

$$(N_{2Am} \times 7,864) + (N_{3Am} \times 8,184) = 1000$$

Resolvendo esse sistema, obtemos 75,416 para N_{2Am}, e 49,723 para N_{3Am}, o que significa uma proporção de 60,27% de mensagens com duas amostras para 39,73% de mensagens com três amostras, o que é comprovado pela Fig. 24.e.

O *buffer* implementado nos MTs suporta até 32 amostras. A frequência de amostragem correspondente a um sistema com um único MT transmitindo 32 amostras a cada leitura é de 1825,650 amostras/s, com cada iteração ocupando 17,528 ms. Como a máxima capacidade de transmissão do MA é de 640 amostras/s, o canal UART entre o MA e o MT é o fator limitante.

Todos os sinais vitais apresentados na seção 2.1 podem ser transmitidos por essa taxa de amostragem. Caso a topologia e os protocolos sejam usados para transmitir sinais biomédicos do tipo *seqüência*, a Tabela 26 mostra os sinais da Tabela 1 compatíveis com as limitações de freqüência de amostragem, enquanto a Tabela 27 mostra os incompatíveis.

Tabela 26 – Sinais biomédicos da Tabela 1 que são compatíveis com o sistema.

Sinal	Freqüência
Eletroretinograma	0,2Hz a 200Hz
Eletro-oculograma	0Hz a 100Hz
Eletroencefalograma de Superfície	0,5Hz a 100Hz
Eletromiografia (superfície)	0,01Hz a 500Hz
Eletrocardiograma	0,05Hz a 100Hz

Tabela 27 – Sinais biomédicos da Tabela 1 que são incompatíveis com o sistema.

Sinal	Freqüência
Potencial de Ação	100Hz a 2kHz
Eletroneurograma	100Hz a 1kHz
Eletrocorticograma	100Hz a 5kHz
Eletromiografia (fibra)	500Hz a 10kHz

5.4 DESENVOLVIMENTOS FUTUROS

O algoritmo do módulo principal foi elaborado preparado para a implementação do tratamento de dados do tipo seqüência. A placa XBEENET possui um conector para cartões *Secure Digital* (SD), já fisicamente conectado ao *hardware* SPI do microcontrolador. Assim, um cartão SD pode ser usado para armazenar as amostras recebidas, e o servidor *web* embarcado ser modificado para recuperar os dados e exibir em forma gráfica.

Com as amostras armazenadas, mesmo os dados do tipo estado e evento podem ser usados para análises, deduzindo-se novas informações. Por exemplo, o dispositivo pode exibir um gráfico de variabilidade de freqüência cardíaca ao longo do tempo.

O protocolo prevê informações sobre o estado da bateria dos MAs, mas um circuito para obter essa informação ainda está por ser desenvolvido.

Por fim, é necessário que se realize um trabalho de campo, em contato com profissionais da área de saúde, para avaliar a satisfatoriedade do sistema em situações reais.

6 CONCLUSÃO

Foi desenvolvido um sistema modular e escalável utilizando microcontroladores de 8 bits, transmissores de rádio e interface Ethernet. Essa combinação se mostrou compatível com o conceito modular e escalável para situações de *home care* e de ambientes hospitalares, para os sinais vitais tradicionais.

O sistema também se mostrou compatível com outros sinais biomédicos, respeitando as condições de número de fontes de dados e taxas de amostragem discutidas.

Um novo protocolo, baseado no protocol I^2C, foi desenvolvido para satisfazer as exigências de comunicação encontradas durante o desenvolvimento. Esse protocolo pode ser utilizado em outros contextos.

REFERÊNCIAS

BRONZINO, J. D. *The Biomedical Engineering Handbook: Second Edition*. [S.l.]: CRC Press LLC, 2000.

CHOI, J. S. et al. The uses of the smartphone for doctors: An empirical study from samsung medical center. *Healthcare Informatics Research*, v. 17, p. 131–138, 2011.

CHUNG, W. *Ubiquitous Healthcare System Based on a Wireless Sensor Network*. Tese (Doutorado) — University of Oulu, Oulu, Finland, 2009.

IEEE. *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements*. New York, NY, USA, 2008.

ISO/IEC. *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*. Case postale 56 – CH-1211 Geneve 20 – Switzerland, 1994.

JACOB, N. A. et al. Low-cost remote patient monitoring system based on reduced platform computer technology. *Telemedicine and e-Health*, v. 17, p. 536–545, 2011.

KIM, J. et al. Development of implementation strategies for u-health services based on the healthcare professionals' experiences. *Telemed J E Health*, v. 17, p. 80–87, 2011.

LEE, S. J. et al. A design of the u-health monitoring system using a nintendo ds game machine. In: *31st Annual International Conference of the IEEE EMBS*. Minneapolis, Minnesota, USA: [s.n.], 2009. p. 1695–1698.

MICROCHIP. *TCP/IP Stack*. 2012. <http://www.microchip.com/tcpip/>. Acessado em 02/01/2012.

MURRAY, E. et al. Why is it difficult to implement e-health initiatives? a qualitative study. *Implementation Science*, v. 6, p. 6, 2011.

OMS. *The World Health Report 1999 – Making a Diference*. 1211 Geneva 27, Switzerland, 1999.

PARK, D. K. et al. Telecare system for cardiac surgery patients: Implementation and effectiveness. *Healthcare Informatics Research*, v. 17, p. 93–100, 2011.

RAUT, C. D.; GIRIPUNJE, V. G. The real-time monitoring system for in-patient based on biomedical data acquisition system. *International Conference on Information and Network Technology*, v. 4, p. 110–114, 2011.

THOMPSON, J. *A Practical Guide to Clinical Medicine*. 2009. <http://meded.ucsd.edu/clinicalmed/vital.htm>. Acessado em 04/06/2013.

TOLENTINO, R. S.; PARK, S. A study on u-healthcare system for patient information management over ubiquitous medical sensor networks. *International Journal of Advanced Science and Technology*, v. 18, p. 1–10, 2010.

W3C. *Web Services Architecture*. [S.l.], 2004.